1 Expirements with Snowflake

author: Raj R., Babu A., and Siva P.
version: 0.1.2
revision: Edition 0

Contents

2 Snowflake & Data Platform Overview

2.1 Snowflake

Snowflake is a cloud based SaaS (Software-as-a-Service) data warehouse solution that makes it possible for organizations to:

- Bring all of the business data together in one place for analysis.
- Allow all analysts, applications or even customers shared access to data, without contention or the need to move or copy data.
- Focus on analyzing data without worrying about hardware, software, or database tuning.
- Utilize the SQL skills we already have, and leverage existing ecosystem of tools from extract transform load or extract load transfrom to business intelligence and more

2.1.1 Snowflake advantages

- Is faster, easier to use, scalable, and far more flexible than other data platforms
- Has an intuitive and simple interface
- Services can be started quickly and delivers high-quality performance
- Solves the concurrency issues with its multi-cluster warehouse architecture
- Loads and processes the data quickly
- Is a fully automated platform
- Supports auto-scaling, big data workload, and data sharing
- Allows a user to scale up the virtual warehouse and provides elasticity as per needs

2.1.2 Snowflake architecture

The Snowflake architecture consists of three layers: storage, compute, and services. Each of these layers can be scaled independently. Organizations can scale up or down the resources as needed and pay only for the resources they consumed.

The 3 main components of the architecture are:

Database Storage — The actual underlying file system in Snowflake is backed by Snowflake's account, all data is encrypted, compressed, and distributed to optimize performance. Snowflake also provides excellent data durability and availability.

Query Processing — Snowflake provides the ability to create Virtual Warehouses which are basically compute clusters that are provisioned behind the scenes. Virtual Warehouses are used to load data or run queries and are capable of doing both of these tasks concurrently. These Virtual Warehouses can be scaled up or down on demand and paused when not in use to reduce the spend on compute.

Cloud Services — Coordinates and handles all other services in Snowflake including sessions, authentication, SQL compilation, encryption, etc.

2.1.3 Connecting to Snowflake

Snowflake provides a few different methods to create database connectivity. One method is to use any of the supported ODBC/JDBC drivers for Snowflake. Web based worksheet can be leveraged within the Snowflake account. SnowSQL CLI is also used to establish connectivity to Snowflake.

2.1.4 Snowflake data features

In the Snowflake platform data can be ingested from a variety of data formats like Parquet, CSV, ORC, JDBC, ODBC, and XML data sources. It supports JSON to a great extent. Furthermore it:

- Provides full support to create views on different data formats.
- Allows data sharing support for securely sharing data with other Snowflake accounts.
- Permits replication and syncing databases across multiple accounts in different regions.
- Enables time travel functions to query the history of data changes over a period of time.

Snowflake is a cloud-agnostic platform providing a unified data management across multiple cloud service providers like Azure, AWS, and GCP. In other words, the Snowflake data warehouse can be built on top of Amazon web services, Microsoft Azure cloud infrastructure, or Google cloud.

The Snowflake command line client - SnowSQL CLI can also access all aspects of managing and using Snowflake. ODBC and JDBC drivers provides easy connectivity from the application/reporting layer to Snowflake.

2.1.5 Setting a Virtual Warehouse

Snowflake defines a virtual warehouse as a cluster of compute resources. The warehouse provides all the required resources like CPU, memory, and temporary storage to start using the Snowflake.

Snowflake provides free trial credit. During free try out registeration, snowflake will create small instance of warehouse for our subscription. This warehouse is setup to auto suspend mode to save credit, if there are no activities.

All data in Snowflake is maintained in databases. Each database consists of one or more schemas. Schemas are logical groupings of database objects like tables, views, functions.

2.2 Data Platform Overview

Modern data architecture allow enterprises to ingest data coming from multiple systems, in a variety of data formats, different speeds, and unknown intervals. Layered data platform design makes it easy to process big data efficiently. It enables organizations to quickly deploy new data analytics business case driven solutions to drive revenue and profitability.

There are four main pillers of modern data platforms

- Data source layer
- Data processing and storage layer
- Analytics layer
- Consumption layer

2.2.1 Data source layer

- Data sources can be inside the enterprise or external
- Data sources generate data in real-time and in batch mode
- A **variety** of data formats can be structured, semi-structured, or unstructured
- The **velocity** (speed of arrival) and **volume** (delivery amount) will differ by sources

2.2.2 Data processing and storage layer

Data processing layer receives data from the data sources, converts the data into a format comprehensible for the data servicing and analytics tool, and stores the data. For example, large amounts of data is stored in the Hadoop distributed file system store (HDFS). Large data processing is performed through Hadoop/Spark system. Data may undergo format changes as it is processed through these systems. Cloud service providers like Amazon, Google, and Microsoft allow a user to build and operate data-centric applications with an infinite scale. Robust and inexpensive storage is fundamental to the operation and scalability of big data architecture.

BigQuery, Azure Synapse,Amazon Redshift, and Snowflake are used as standalone solutions for big data processing or in combination of Hadoop/Spark ecosystems.

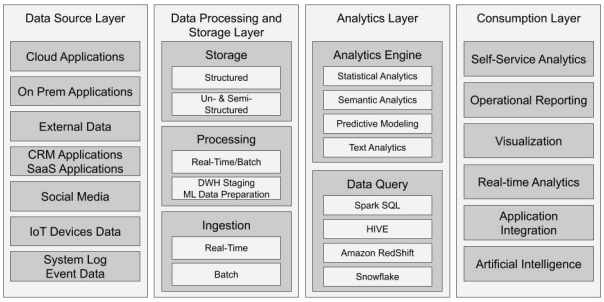
2.2.3 Analytics layer

The Analytics layer reads the data ingested and transformed by the data processing and storage layer of big data ecosystem. This layer consists varieties of data analytic tools for different user requirements. This layer provides the data discovery mechanisms from huge volumes of data. Apache spark SQL, Hive, Apache spark streaming, Machine learning libraries, Apache spark GraphX, SQL libraries, and number of other tool sets are utilized in this layer to understand underlying data landscape.

2.2.4 Consumption layer

Consumption layer is also called the business intelligence layer. This layer receives results from the Analytical layer and presents the results using visualization tools, and business processes.

The following diagram summarize layers in data management solution



2.3 Summary

Capabilities in the Snowflake enable data lake, data share, data visualization, data science, and data engineering for the organizations We will be working on the real world business case from the input data source to the visualization using Snowflake data platform.

This book includes:

- Data Sources
- Different source data formats, available to build platform
- Snowflake database development
- Snowflake data transformation
- Data pipeline orchestration
- Data lineage, pipeline audit and control
- SnowSQL and JavaScript based stored procedures

- Business use cases, users are tasked to solve
- Data visualization

3 Data Load Software

Snowflake data load can be achieved

- SnowSQL - CLI allows bulk load data using SQL commands
- Snowpipe is utilized to automate large data set across multiple platforms
- Web interfaces are used to load small amounts of data
- Snowflake works with a wide range third parties like ETL/ELT Informatica, Talend, and Azure Data Factory

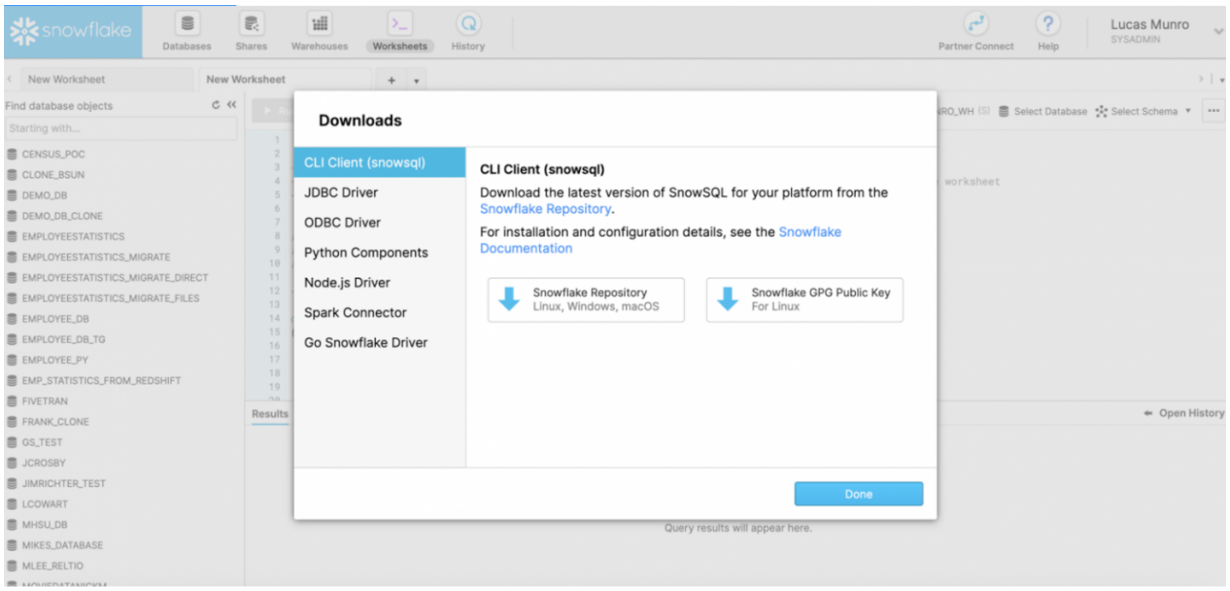
3.1 SnowSQL UI

The Snowflake web interface provides a convenient wizard for loading limited amounts of data into a table from a small set of flat files. The Web UI allows us to simply select the table we want to load and by clicking the LOAD button we can easily load a limited amount of data into Snowflake. The wizard simplifies loading by combining the staging and data loading phases into a single operation and it also automatically deletes all the staged files after loading.

3.2 SnowSQL CLI

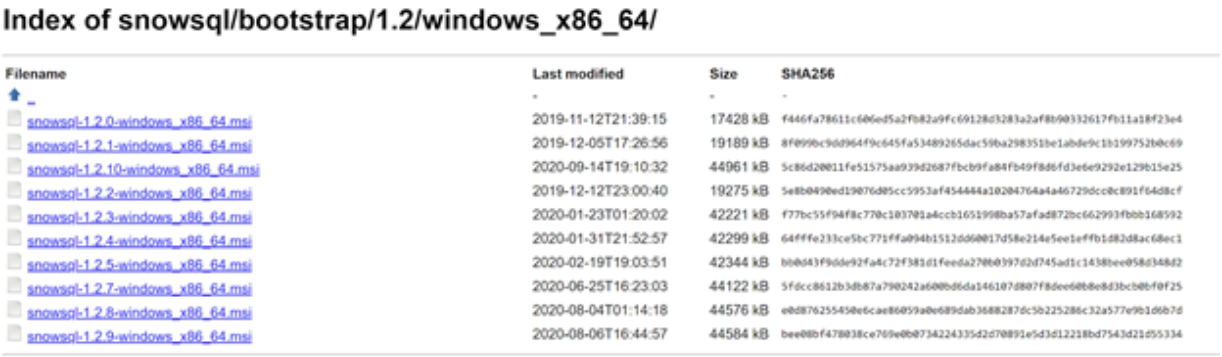
SnowSQL CLI is the command line client for Snowflake. It executes SQL queries and performs all DDL and DML operations. It is an easy way to access Snowflake from the command line. It has the same capabilities as the Snowflake UI.

Login into Snowflake and click on help in the top right corner and download



We can also download SnowSQL CLI from the location URL :

https://Snowflakesolutions.net/snow-sql-cliclient/



Index of snowsql/bootstrap/1.2/windows_x86_64/

Filename	Last modified	Size	SHA256
_	-	-	-
snowsql-1.2.0-windows_x86_64.msi	2019-11-12T21:39:15	17428 kB	f446fa78611c606ed5a2fb82a9fc69128d3283a2af8b90332617fb11a18f23e4
snowsql-1.2.1-windows_x86_64.msi	2019-12-05T17:26:56	19189 kB	8f099bc9dd964f9c645fa53489265dac59ba2983511be1abde9c1b199752b0c69
snowsql-1.2.10-windows_x86_64.msi	2020-09-14T19:10:32	44961 kB	5c86d20011fe51575aa939d2687fbcb9fa84fb49f8d6fd3e6e9292e129b15e25
snowsql-1.2.2-windows_x86_64.msi	2019-12-12T23:00:40	19275 kB	5e8b0490ed19076d05cc5953af454444a10204764a4a46729dcc0c891f64d8cf
snowsql-1.2.3-windows_x86_64.msi	2020-01-23T01:20:02	42221 kB	f77bc55f94f8c770c103701a4ccb1651998ba57afad872bc662993fbbb168592
snowsql-1.2.4-windows_x86_64.msi	2020-01-31T21:52:57	42299 kB	64fffe233ce5bc771ffa094b1512dd60017d58e214e5ee1effb1d82d8ac68ec1
snowsql-1.2.5-windows_x86_64.msi	2020-02-19T19:03:51	42344 kB	bb0d43f9dde92fa4c72f381d1feeda2700397d2d745ad1c1438bee058d348d2
snowsql-1.2.7-windows_x86_64.msi	2020-06-25T16:23:03	44122 kB	5fdcc8612b3db87a790242a600bd6da146107d807f8dee60b8e8d3bcb0bf0f25
snowsql-1.2.8-windows_x86_64.msi	2020-08-04T01:14:18	44576 kB	e0d876255450e6cae86059a0e689dab3688287dc5b225286c32a577e9b1d6b7d
snowsql-1.2.9-windows_x86_64.msi	2020-08-06T16:44:57	44584 kB	bee08bf470038ce769e0b0734224335d2d70891e5d3d12218bd7543d21d55334

SnowSQL CLI gets invoked with snowsql.exe

SnowSQL Command Usage

We can use the following command to connect to Snowflake using SnowSQL.

```
snowsql -a accountName -u userName -d databaseName -s schemaName
```

For example, consider the following command to connect Snowflake from Windows command line.

```
snowsql -a xxxxxx.us-east-1 -u xxxxxx -d demo_db -s public
Password:******
```

3.3 Snowpipe

Snowpipe is a mechanism to load high frequency or streaming data. Snowpipe provides the capability to load the data as soon as it becomes available in a defined stage. It can load a near real-time or micro-batch load of data. Snowpipe is a serverless architecture and does not use virtual warehouse resources. Snowpipe has its own resources, and those are managed by Snowflakes instances.

Snowpipe makes REST API available to load data from

- Amazon Web Services (AWS) -Amazon S3
- Google Cloud Platform - Cloud Storage
- Microsoft Azure Blob Storage, Data Lake Storage Gen2
- Hadoop and Spark big data platform

4 Snowflake Staged Area

Snowflake allows several types of staged file areas. A stage is a temporary storage area in the Snowflake warehouse.

Snowflake supports two types of stages for storing data files- external and internal.

External stages are storage locations outside the Snowflake environment in another cloud storage location. It could be Amazon S3 storage, Microsoft Azure storage, or Google Cloud Storage buckets. This provides greater flexibility to accessing the data in Snowflake.

The internal stage stores data files internally within Snowflake. Internal stages can be either permanent or temporary.

Internal storage is classified into three categories.

4.1 User Stage

Each Snowflake user is attached to the default stage to store the file. User Stage is only accessible by specific user.

```
put file://D:\Snowflake\sample_file.csv @~/staged;
```

The file will be uploaded to the user stage and a user can verify the same using LIST command.

```
list @~/staged;
```

The user stages are referenced using @~.

For example, use LIST @~ to list the files in a user stage

4.2 Table Stage

Each table has a Snowflake stage allocated for storing files. Multiple users can access a single table stage area.

```
put file://D:\Snowflake\sample_file.csv @%test;
```

Users can also specify the subfolder within the table stage to upload the files.

```
put file://D:\Snowflake\sample_file.csv @%test/sample_csv/;
```

The table stages are referenced using **@%tableName**.

For example, to list the files in a table stage

```
LIST @%test
```

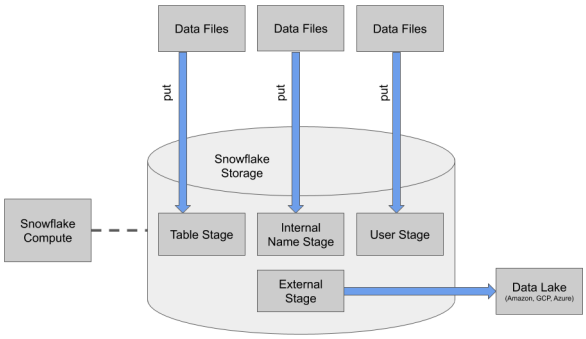
Table stages have the same name as the table. The table stages cannot be altered or dropped.

Table stages do not support setting file format options. Instead, they specify file format details in the COPY command.

4.3 Internal Named Stage

Internal stages are named database objects. Internal Named stages are the recommended stage to load the tables. It is accessible by multiple users. Internal stage can load multiple tables. File formats can be specified while creating an Internal Named stage.

Internal storage copies data within the Snowflake warehouse as shown below.



5 Project Data Source

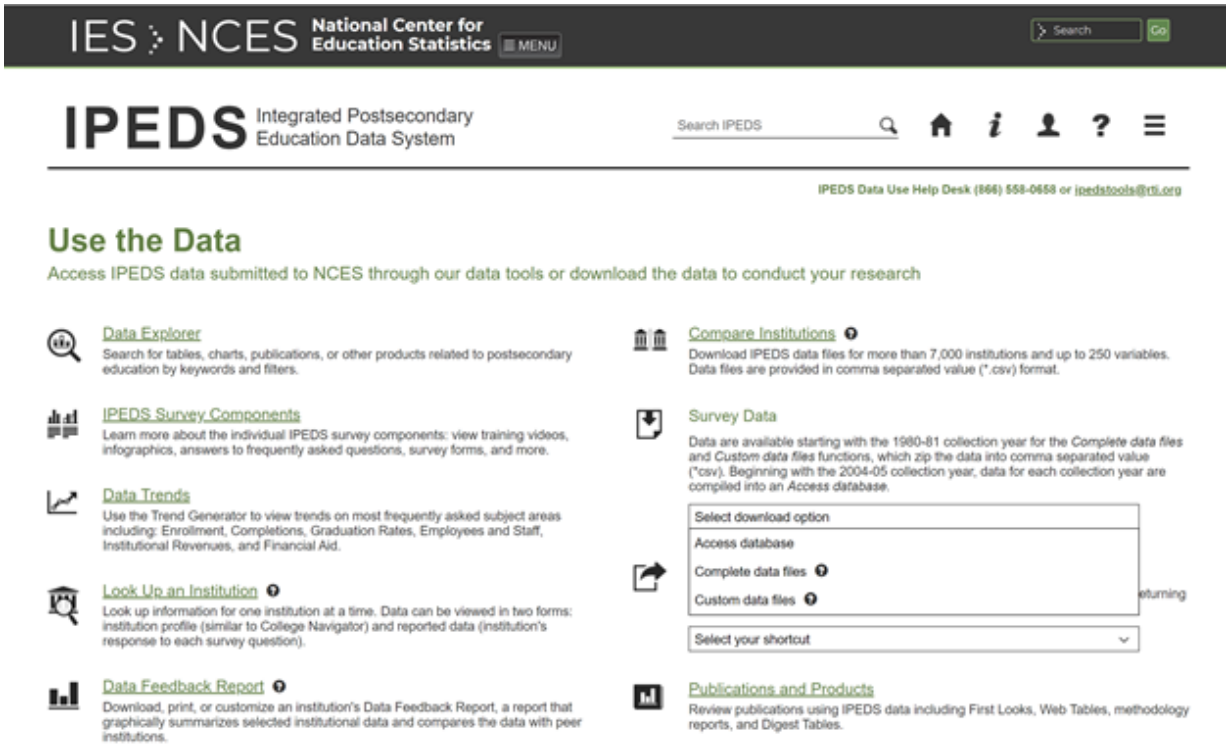
The Institute of Education Sciences (IES) is the statistics, research, and evaluation arm of the U.S. Department of Education. IES is independent and non-partisan. IES mission is to provide scientific evidence on which to ground education practice and policy and to share this information in formats that are useful and accessible to educators, parents, policymakers, researchers, and the public.

This book is designed to use data from IES and build analytics engine to answer business questions for the corporation.

Integrated Postsecondary Education Data System (IPEDS) - URL : https://nces.ed.gov/ipeds/ is the primary source from IES for the information on U.S. colleges, universities, and technical and vocational institutions.

- We are tasked to ingest variety of data formats
- Tranform data for business users
- Build data pipeline to process data on perodic basis
- Create descriptive analytics around the use cases laid out by the business

All of the data used in this book is downloaded from - URL : https://nces.ed.gov/ipeds/use-the-data



Data is downloaded for the three academic years (2017, 2018, and 2019).

Disclaimer – We have modified this data to explain Snowflake functionality. Analytics results in this book are used to showcase and learn Snowflake features. **Analytics results from this book should not be used to perform research on the institutions. Please refer to IES website for statistics and research for Academic Institutions.**

5.1 Data flow process

For this project in the book, we will be following four building block of data transformation layers.

5.1.1 Staged file

All the data from the source system will be brought into the internal Staged file location of the Snowflake. There is no transformation applied on this dataset. It is the purest form data from the transactional system and existing big data platforms.

5.1.2 Operation datastore (OD)

These are first level of physical table created in the Snowflake. Data from staged file is transformed to very basic minimum needs. OD table structure is very close resembles of staged file with proper data type and column names. Additional columns to help track on data lineage.

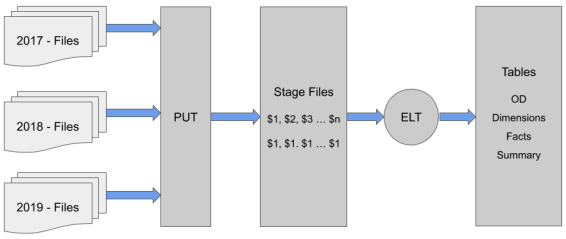
5.1.3 Servicing layer entities

This is the final destination for transformed data available for business intelligence layer. OD tables are used as input source to populate the servicing layer tables.

5.1.4 Dataflow Summary

Not all data from OD tables may land up in the servicing layer. Normally business users would want to use the tools like PowerBI, Tableau, and perform raw queries on the dataset. Best place for the business users to perform these analytics is the service layer. Data Science may need data attributes not in the service layer to build predictive analytics. They can be granted access to OD tables and service layer entities. Data engineering needs to create transformation, perform meta data management, keep track of data pipelines so the access should be granted to the all areas of data flow entities.

The below diagram shows the data flow process between files and the staging table. In this case, the stored procedures are used to load data from staged files to staging tables.



5.2 Book Datasource

The files used in this book are:

5.2.1 Institutional Characteristics files [HDR]

This file contains directory information for every institution in the IPEDS universe. It includes name, address, city, state, zip code and various URL links to the institution's home page, admissions, financial aid offices and the net price calculator.

5.2.2 12-Month Enrollment [EFFY]

This file contains the unduplicated headcount of students enrolled over a 12-month period for both undergraduate and graduate levels. Each record is uniquely defined by the variables IPEDS ID, and the level of enrollment.

5.2.3 Admissions and Test Scores [ADM]

This file contains information about the undergraduate selection process for entering first-time, degree/certificate-seeking students.

5.2.4 Student charges for academic year programs [IC*AY]

This file contains data on student charges for a full academic year. The price of attendance includes amounts for published tuition and required fees, books and supplies, room and board and other expenses.

5.2.5 Code Mapping Data

This is semi structured dataset with special character delimeter. One file holds data for multiple destination table. Based on the value of the first column in the file, destination table is identified.

5.3 Data collection notes

- Original text delimited data in
- EFFY file is converted to JSON format
- ADM file is converted to Parquet format
- IC*AY file is converted to ORC format
- There is no change in the HDR file format (CSV format)

> ***Note***
>
> Utilities used to convert the original CSV files into different data formats are available to download at: https://github.com/versatiledp/ExperimentsSnowflake/tree/main/utilities
>
> All the datafiles used throughout the books are available to download at:
>
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/data/input

After downloading the files, copy to the folder C:\Snowflake\data\input. The files in this book are loaded from local folder (like C:\Snowflake\data\input*.*) to the Snowflake stage area using SnowSQL CLI.

5.4 Summary

We receive data files in various formats like CSV, ORC, Parquet, and JSON. These files are available in the folders by year. They are brought into the Snowflake staged files using the PUT command. Each staged file contains the entire data for that specific year. From these staged files, OD tables are populated. OD tables are physical tables in a Snowflake database. Using these OD tables, servicing layer tables of data-warehouse are populated. Snowflake stored procedures, views and functions are used for ETL/ELT processes.

6 Ingesting Snowflake Stage

6.1 Internal stage area for IPEDS

We are going to load different file formats like CSV, JSON, Parquet, and ORC. Internal named stage needs to be created for each of the file formats before loading data.

6.2 Create internal stage area IPEDS

Create the file formats and stage area for the text file

1. tab delimited
2. comma separated with no header
3. comma separated with header

```
-- Number of header lines at the start of the file.
-- The COPY command skips header lines when loading data
-- All these file formats can be used to create different stage areas
CREATE OR REPLACE  FILE FORMAT
                ff_IPEDS_CSVSkipHeaderTabDelimited
                        TYPE =   CSV
                        FIELD_DELIMITER = '\t'  COMPRESSION = AUTO
                        SKIP_HEADER = 1;
CREATE OR REPLACE  FILE FORMAT
                ff_IPEDS_CSVSkipHeaderCommaDelimited
                        TYPE =   CSV
                        FIELD_DELIMITER = ','  COMPRESSION = AUTO
                        SKIP_HEADER = 1;
-- The COPY command does not skip any lines.
CREATE OR REPLACE  FILE FORMAT
                ff_IPEDS_CSVHeaderTabDelimited
                        TYPE =   CSV
                        FIELD_DELIMITER = '\t'  COMPRESSION = AUTO
                        SKIP_HEADER = 1 ;
CREATE OR REPLACE  FILE FORMAT
                ff_IPEDS_CSVHeaderCommaDelimited
                        TYPE =   CSV
                        FIELD_DELIMITER = ','  COMPRESSION = AUTO
                        SKIP_HEADER = 0 encoding = 'iso-8859-1'
                        FIELD_OPTIONALLY_ENCLOSED_BY='"';
-- create satge area to load csv files for the project
CREATE OR REPLACE STAGE IPEDS_HD FILE_FORMAT
                        = ff_IPEDS_CSVHeaderCommaDelimited;
CREATE OR REPLACE STAGE IPEDS_CM FILE_FORMAT
                        = ff_IPEDS_CSVSkipHeaderTabDelimited;
```

--- Create JSON file format & stage area

```
-- Create file format to load json file
CREATE OR REPLACE FILE FORMAT ff_IPEDS_Json
                TYPE =JSON TRIM_SPACE = TRUE;
-- Create stage area to load json
CREATE OR REPLACE stage IPEDS_EFFY FILE_FORMAT = ff_IPEDS_Json;
```

--- Create ORC file format & stage area

```
-- Create file format to load ORC file
CREATE OR REPLACE FILE FORMAT ff_IPEDS_ORC
                        TYPE =ORC TRIM_SPACE = TRUE;
-- Create stage area to load ORC file
CREATE OR REPLACE stage IPEDS_IC FILE_FORMAT = ff_IPEDS_ORC;
```

--- Create Parquet file format & stage area

```
-- Create Parquet file format
CREATE OR REPLACE  FILE FORMAT ff_IPEDS_Parquet
                        TYPE =PARQUET TRIM_SPACE = TRUE;
-- Create stage area to load Parquet file
CREATE OR REPLACE stage IPEDS_ADM FILE_FORMAT = ff_IPEDS_Parquet;
```

Once these formats are created, we can check the staged area and formats to load Snowflake internal stage area with

```
SHOW STAGES ;
SHOW STAGES LIKE '%IPED%';
```

Row	created_on	name	database_name	schema_name
1	2020-12-13 ...	IPEDS_ADM	IPEDS	PUBLIC
2	2020-12-13 ...	IPEDS_CM	IPEDS	PUBLIC
3	2020-12-13 ...	IPEDS_EFFY	IPEDS	PUBLIC
4	2020-12-13 ...	IPEDS_HD	IPEDS	PUBLIC
5	2020-12-13 ...	IPEDS_IC	IPEDS	PUBLIC

```
SHOW FILE FORMATS;
SHOW FILE FORMATS LIKE '%IPED%';
```

Row	created_on	name	database_name	schema_name	type	owner	comment	format_options
1	2020-12-13 19:26:15...	FF_IPEDS_CSVHEAD...	IPEDS	PUBLIC	CSV	SYSADMIN		{"TYPE":"CSV","REC...
2	2020-12-13 19:26:14...	FF_IPEDS_CSVHEAD...	IPEDS	PUBLIC	CSV	SYSADMIN		{"TYPE":"CSV","REC...
3	2020-12-13 19:26:14...	FF_IPEDS_CSVSKIPH...	IPEDS	PUBLIC	CSV	SYSADMIN		{"TYPE":"CSV","REC...
4	2020-12-13 19:26:14...	FF_IPEDS_CSVSKIPH...	IPEDS	PUBLIC	CSV	SYSADMIN		{"TYPE":"CSV","REC...
5	2020-12-13 19:26:5...	FF_IPEDS_JSON	IPEDS	PUBLIC	JSON	SYSADMIN		{"TYPE":"JSON","FILE...
6	2020-12-13 19:27:3...	FF_IPEDS_ORC	IPEDS	PUBLIC	ORC	SYSADMIN		{"TYPE":"ORC","TRIM...
7	2020-12-13 19:28:0...	FF_IPEDS_PARQUET	IPEDS	PUBLIC	PARQUET	SYSADMIN		{"TYPE":"PARQUET","...

Summary

We have created an internal named stage in Snowflake to ingest CSV files, Parquet file, ORC file, and JSON files.

We will use the following stage area to load the raw files.

- IPEDS_ADM
- IPEDS_EFFY
- IPEDS_CM
- IPEDS_HD
- IPEDS_IC

6.3 Load the files to internal stage for IPEDS

Connect to the Snowflake database using SnowSQL CLI and load the following files to the staging area.

Load academic institution (HDR) files - These are CSV files

```
put file://C:\snowflake\data\input\HD2017.csv @IPEDS_HD;
put file://C:\snowflake\data\input\HD2018.csv @IPEDS_HD;
put file://C:\snowflake\data\input\HD2019.csv @IPEDS_HD;
```

Load enrollment (EFFY) files - These are JSON files

```
put file://C:\snowflake\data\input\effy2017_rv.json @IPEDS_EFFY;
put file://C:\snowflake\data\input\effy2018_rv.json @IPEDS_EFFY;
put file://C:\snowflake\data\input\effy2019_rv.json @IPEDS_EFFY;
```

Load institutional (IC) charges files - These are ORC files. All the partitioned are loaded into specific folder of ORC staging area.

```
put file://C:\snowflake\data\input\ic2017_ay_orc\*.* @IPEDS_IC/2017/;
put file://C:\snowflake\data\input\ic2018_ay_orc\*.* @IPEDS_IC/2018/;
put file://C:\snowflake\data\input\ic2019_ay_orc\*.* @IPEDS_IC/2019/;
```

Load admission files - These are Parquet files. All the partitioned are loaded into specific folder of Parquet staging area.

```
put file://C:\snowflake\data\input\adm2017.parquet\*.* @IPEDS_ADM/2017/;
put file://C:\snowflake\data\input\adm2018.parquet\*.* @IPEDS_ADM/2018/;
put file://C:\snowflake\data\input\adm2019.parquet\*.* @IPEDS_ADM/2019/;
```

Load code mapping files - These are CSV files.

```
put file://C:\snowflake\data\input\CodeMappingData.txt @IPEDS_CM;
```

Confirm stage file availability in Snowflake

Once these files are loaded to the stage area, we can check the availability of these data files in the internal stage area as below:

```
list @IPEDS_HD;
```

Row	name	size	md5
1	ipeds_hd/HD2017.csv.gz	1148288	ca4cc4a7a0ea364ef6ab6f1a838f9774
2	ipeds_hd/HD2018.csv.gz	1116640	f1b879ee545c8dcd838470bb0031d5a7
3	ipeds_hd/HD2019.csv.gz	1082432	c8ee97d50592a5d09a53e3c84ef1dca9

```
list @IPEDS_EFFY;
```

Row	name	size	md5
1	ipeds_effy/effy2017_rv.json.gz	863072	980c9dea3a4f59f4b6694a13a1d07022
2	ipeds_effy/effy2018_rv.json.gz	847584	78a82bca3b0eb0cff1590c2fe7005ca4
3	ipeds_effy/effy2019_rv.json.gz	821344	99b9dc5cca1eb4ddf5643b23a6c032cf

```
list @IPEDS_IC;
```

Row	name	size	md5
1	ipeds_ic/2017/part-00000-8372409f-8fed-46...	1298704	34a4a454b34de8c07216ab2ed7279e92
2	ipeds_ic/2018/part-00000-1adbf580-7a8c-41...	1273200	75cac6ed3920bb8a28655e22cccdb8b9
3	ipeds_ic/2019/part-00000-5862ce87-a5f7-48...	1253696	2b370250159987edd5e9f109f101fe81

```
list @IPEDS_ADM;
```

Row	name	size	md5
1	ipeds_adm/2017/part-00000-9f13e205-1455-...	158128	5f2bc0de95ae6016c000dad9d74252a1
2	ipeds_adm/2018/part-00000-17bcf105-a5eb-...	156400	c25c803b8a16634d99d8b6301f3711c8
3	ipeds_adm/2019/part-00000-3f086fd7-4ecb-...	153360	44c910c413bcafe027b30676631ab424

```
list @IPEDS_CM;
```

Row	name	size	md5
1	ipeds_cm/CodeMappingData.txt.gz	38688	9fbee9ad38240bdbfb4b0928d8743bb3

All the files from data source layers are ingested into the Snowflake staged area.

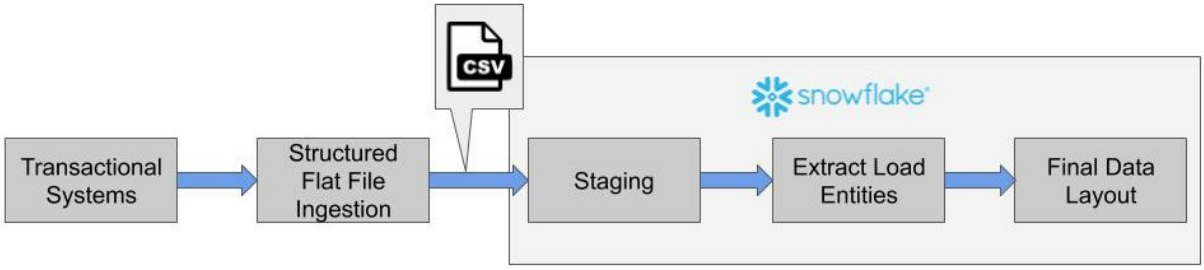
If there are any issues in stage area, drop it using **remove @IPEDS_ADM**

> ***Note***
>
> Code available to download at
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/source/code/SQL/StagedFile

7 Attribute Mapping

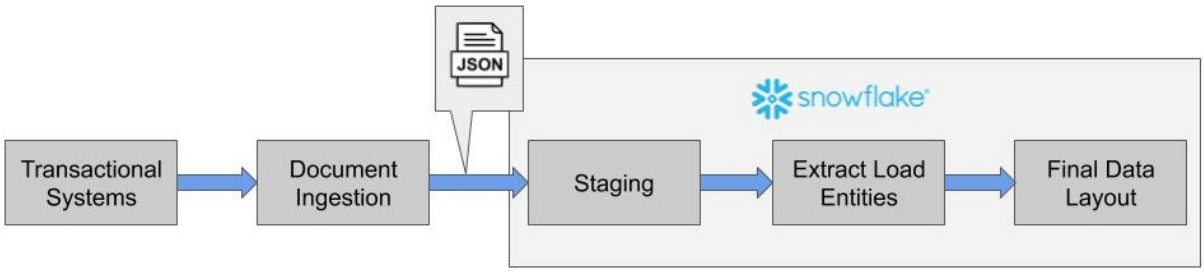
7.1 Academic Institution

Institutional Characteristics Header files [HDR] are in CSV format. These files are directly ingested into Snowflake staging from the source system. ELT layer defines the schema for the CSV file with data types. The following diagram displays mapping between the CSV attribute and the stage layer. The OD layer is used to process data for destination layer with additional transformations.



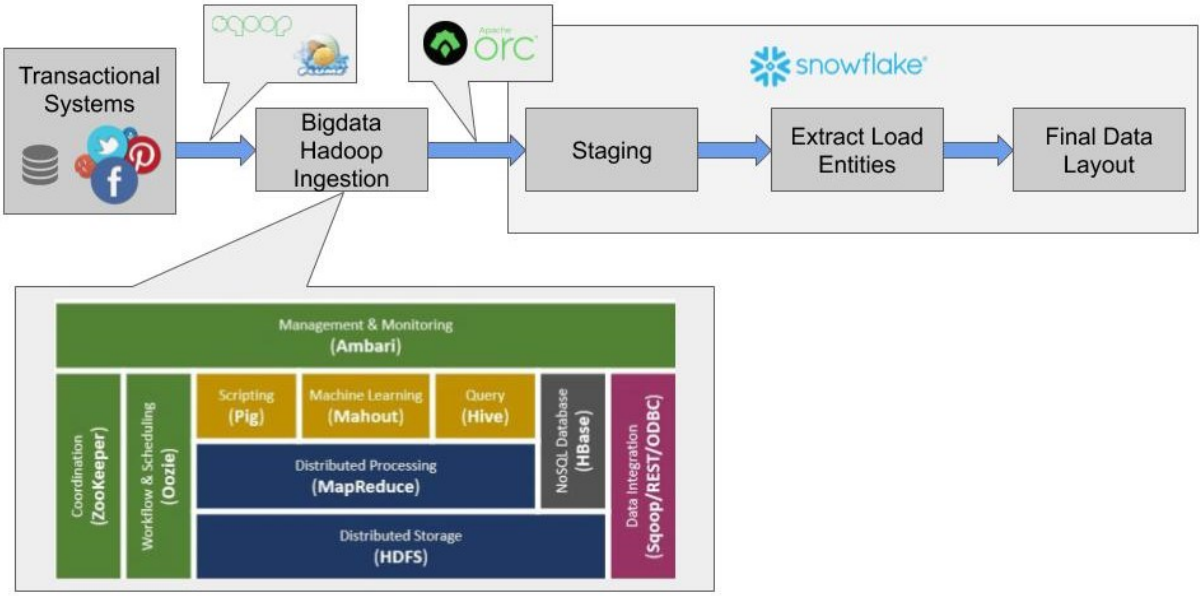
7.2 Enrollment

Enrollment files [EFFY] are in JSON format. JSON is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types. It is a very common data format, with a diverse range of applications, such as serving as a replacement for XML in AJAX systems. In our use case – we are assuming Enrollment files are ingested (EFFY files) in the JSON format directly to Snowflake. ELT layer defines the mapped attributes for JSON attributes with data types. The following diagram displays mapping between JSON attribute and stage layer. The Stage layer is used to process data for the destination layer with additional transformation.



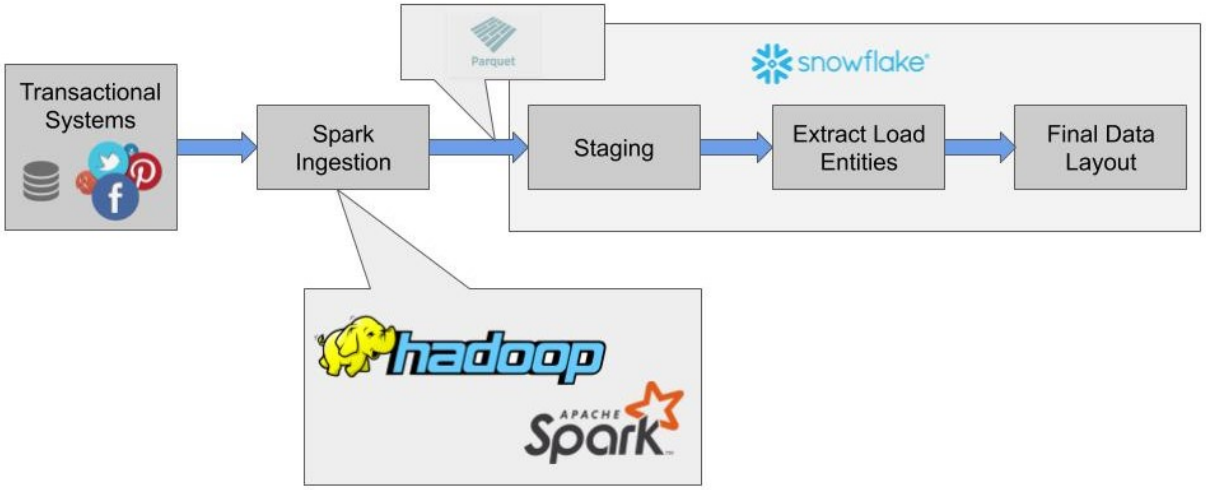
7.3 Institutional Charges

Institutional Charges [IC] dataset is in the ORC formt. The Optimized Row Columnar (ORC) file format provides a highly efficient way to store Hive data. Using ORC files improves performance when Hive is reading, writing, and processing data. It ideally stores data in compact form and enables skipping over irrelevant parts without the need for large, complex, or manually maintained indices. In our use case, – we are assuming Institutional Charge text files are ingested (ICAY files) in HDFS. They are processed through Hive. Output of the big data processing (ORC files) are used to load in Snowflake. ELT layer defines the schema for ORC file with data types. The following diagram displays mapping between the ORC attribute and the stage layer. The stage layer is used to process data for the destination layer with additional transformations and normalization.



7.4 Admission data

Admission data [ADM] files are available in the Parquet data format. Parquet, an open-source file format for Hadoop stores nested data structures in a flat columnar format. Compared to a traditional approach where data is stored in a row-oriented approach, Parquet is more efficient in terms of storage and performance. It is especially useful for queries that read specific columns from a wide table. Parquet provides optimizations to speed up queries. Parquet is a far more efficient file format than CSV or JSON, supported by many data processing systems. Spark SQL provides support for both reading and writing Parquet files that automatically capture the schema of the original data. The following diagram lays out the architecture from data source to Parquet file. In our use case – we are assuming - IPEDs admission data text files are ingested (ADM files) in HDFS and processed through the Spark. Output of the spark processing (Parquet files) are used to process in Snowflake. ELT layer defines the schema for Parquet file with data types. The following diagram displays mapping between the Parquet attribute and the stage layer. The stage layer is used to process data for the final destination layer with additional transformations.



7.5 Reference/lookup data

Code value lookup files are CSV formatted. These files are directly ingested into Snowflake staging from source system. ELT layer defines the schema for the CSV file with data types.

7.6 Summary

Data from the source systems goes through multiple data processing layers in transactional and big data platforms. That results into varieties of data files in the multiple data formats. We need to map those data files into analytical meta data layer. This chapter summarize all data formats we will be dealing in the processing layer of analytical data platform.

> ***Note***
>
> Mapping between the source system and Snowflake meta data is available in appendix. It is also available to view at:
>
> https://versatiledp.github.io/ExperimentsSnowflake/Mapping/AcademicInstitute.htm
> https://versatiledp.github.io/ExperimentsSnowflake/Mapping/Enrollment.htm
> https://versatiledp.github.io/ExperimentsSnowflake/Mapping/InstitutionCharges.htm
> https://versatiledp.github.io/ExperimentsSnowflake/Mapping/AdmissionStat.htm
> https://versatiledp.github.io/ExperimentsSnowflake/Mapping/CodeValue.htm

8 CSV Ingestion

8.1 Academic Institution

The academic institution file contains the high level information about institutions in the country by year. The following files contain the information about the institution for the year 2017, 2018 and 2019 respectively.

- HD2017.csv
- HD2018.csv
- HD2019.csv

We will learn how to

- Use ingested CSV files from the staged Snowflake region
- Transform data using Snowflake JavaScript stored procedure
- Call stored procedure
- Build views in Snowflake
- Manage schema drifts from one file to another file
- Build processes so schema drift does not create impdiments with old historical data
- Combine column attributes to JSON objects as part of transformation
- Perform SQL MERGE using hash key

In staged files, data can be located in an internal stage (in a Snowflake database) or an external named stage (Amazon S3, Google Cloud Storage, or Microsoft Azure). Staged files are identified by the @ character prefixed to the name. Academic institution data files are received in CSV format and a staged file is created.

In this case, **@IPEDS_HD** is the staged file containing information about the Academic institutions. From the staged files which are created out of CSV format, data is retrieved using SQL command SELECT $<column_number>. The **<column_number>** represents the order in which the columns are available in the text file starting with the number 1. In other words, $1 refers to first column values, $2 refers to second column values, and $n refers to nth column values from the staged file. For example: SELECT $1, $2 FROM @IPEDS_HD; The above command returns the first and second column values of the Institution file.

Here is the output from @IPEDS_HD using the SQL command. Header record from the input file is presented as data record:

```
SELECT $1,$2,$3,$4,$5,$6,$7,……..,$71,$72 FROM @IPEDS_HD;
```

Row	$1	$2	$3	$4	$5	$6	$7	$71
1	UNITID	INSTNM	IALIAS	ADDR	CITY	STABBR	ZIP	DFRCG
2	100654	Alabama A & M U...	AAMU	4900 Meridian St...	Normal	AL	35762	122
3	100663	University of Alab...		Administration Bl...	Birmingham	AL	35294-0110	109
4	100690	Amridge University	Southern Christia...	1200 Taylor Rd	Montgomery	AL	36117-3553	141
5	100706	University of Alab...	UAH |University o...	301 Sparkman Dr	Huntsville	AL	35899	112
6	100724	Alabama State U...		915 S Jackson St...	Montgomery	AL	36104-0271	131
7	100733	University of Alab...		500 University Bl...	Tuscaloosa	AL	35401	-2
8	100751	The University of ...		739 University Blvd	Tuscaloosa	AL	35487-0166	111
9	100760	Central Alabama ...		1675 Cherokee Rd	Alexander City	AL	35010	74
10	100812	Athens State Uni...		300 N Beaty St	Athens	AL	35611	151

To ignore header records from loading into the data table, we can skip the header information row in staged file.

```
SELECT $1,$2,$3,$4,$5,$6,$7,……..,$71,$72
FROM @IPEDS_HD WHERE metadata$file_row_number > 1;
```

Row	$1	$2	$3	$4	$5	$6	$7	$71	$72
1	100654	Alabama A & M U...	AAMU	4900 Meridian St...	Normal	AL	35762	34.783368	112
2	100663	University of Alab...		Administration Bl...	Birmingham	AL	35294-0110	33.505697	98
3	100690	Amridge University	Southern Christia...	1200 Taylor Rd	Montgomery	AL	36117-3553	32.362609	129
4	100706	University of Alab...	UAH University of...	301 Sparkman Dr	Huntsville	AL	35899	34.724557	102
5	100724	Alabama State U...		915 S Jackson St...	Montgomery	AL	36104-0271	32.364317	120
6	100733	University of Alab...		500 University Bl...	Tuscaloosa	AL	35401	33.207015	-2
7	100751	The University of ...		739 University Blvd	Tuscaloosa	AL	35487-0100	33.211875	96
8	100760	Central Alabama ...		1675 Cherokee Rd	Alexander City	AL	35010	32.924780	71
9	100812	Athens State Uni...		300 N Beaty St	Athens	AL	35611	34.806793	139

The attributes from the the staged file do not hold column name and column types. We need to provide schema on read for each of the $1, $2,,, $n columns in the staged file @IPEDS_HD. This is accompolished using mapping documents between staged layer and column names in OD tables.

8.2 OD table for Academic Institution

Operational data (OD) tables are physical tables and populated from the corresponding staged files. ELT process will copy data from Staged files with very little transformation and no metadata to OD tables and descriptive column names. We can add more columns to the OD table if necessary. We have added 3 additional columns like AcademicYear, IngestedFileName, and RowNumber to the OD table for using them in the ETL/ELT process and audit purposes.

Once the column mapping is clearly defined, we have to populate our operational data table for academic institutions called **od_AcademicInstitution** out of this staged file **@IPEDS_HD**.

Here is the SQL command to create an OD table **od_AcademicInstitution**.

```
DROP TABLE IF EXISTS od_AcademicInstitution;

CREATE TABLE od_AcademicInstitution (
  InstitutionIdentifier INTEGER,
  InstitutionName STRING,
  InstitutionNameAlias STRING,
  StreetAddress STRING,
```

```
City STRING,
State VARCHAR(10),
ZipCode VARCHAR(20),
StateCode INTEGER,
EconomicAnalysisRegions INTEGER,
ChiefAdministrator STRING,
ChiefAdministratorTitle STRING,
TelephoneNumber STRING,
EmployerIdentificationNumber STRING,
DunBradstreetNumbers STRING,
PostsecondaryEducationIDNumber STRING,
TitleIVEligibilityIndicatorCode INTEGER,
InstitutionsWebAddress STRING,
AdmissionsOfficeWebAddress STRING,
FinancialAidOfficeWebAddress STRING,
OnlineApplicationWebAddress STRING,
NetPriceCalculatorWebAddress STRING,
VeteransMilitaryServiceTuitionPoliciesWebAddress STRING,
StudentRightAthleteGraduationRateWebAddress STRING,
DisabilityServicesWebAddress STRING,
SectorOfInstitution INTEGER,
LevelOfInstitution INTEGER,
ControlOfInstitution INTEGER,
HighestLevelOfOffering INTEGER,
UndergraduateOffering INTEGER,
GraduateOffering INTEGER,
HighestDegreeOffered INTEGER,
DegreeGrantingStatus INTEGER,
HistoricallyBlackCollegeOrUniversity INTEGER,
InstitutionHasHospital INTEGER,
InstitutionGrantsMedicalDegree INTEGER,
TribalCollege INTEGER,
DegreeOfUrbanization INTEGER,
InstitutionOpenToGeneralPublic INTEGER,
StatusOfInstitution VARCHAR(10),
UnitidForMergedSchools STRING,
YearInstitutionWasDeletedFromIPEDS INTEGER,
DateInstitutionClosed VARCHAR(20),
InstitutionIsActive INTEGER,
PrimarilyPostsecondaryIndicator INTEGER,
PostsecondaryInstitutionIndicator INTEGER,
PostsecondaryAndTitleIvInstitutionIndicator INTEGER,
ReportingMethodForStudentCharges INTEGER,
InstitutionalCategory INTEGER,
CarnegieClassification2015Basic INTEGER,
CarnegieClassification2015UndergraduateProgram INTEGER,
CarnegieClassification2015GraduateProgram INTEGER,
CarnegieClassification2015UndergraduateProfile INTEGER,
CarnegieClassification2015EnrollmentProfile INTEGER,
CarnegieClassification2015SizeSetting INTEGER,
CarnegieClassification20052010Basic INTEGER,
CarnegieClassification2000 INTEGER,
LandGrantInstitution INTEGER,
InstitutionSizeCategory INTEGER,
MultiCampusOrganization INTEGER,
```

```
  NameOfMultiCampusOrganization STRING,
  IdentificationNumberOfMultiCampusOrganization INTEGER,
  CoreBasedStatisticalArea INTEGER,
  CBSATypeMetropolitanMicropolitan INTEGER,
  CombinedStatisticalArea INTEGER,
  NewEnglandCityAndTownArea INTEGER,
  FIPSCountyCode INTEGER,
  CountyName STRING,
  StateAnd114thCongressionalDistrictID INTEGER,
  LongitudeLocation NUMBER(32, 0),
  LatitudeLocation NUMBER(32, 0),
  NCESGroupCategory INTEGER,
  DataFeedbackReport INTEGER,
  AcademicYear INTEGER,
  IngestedFileName STRING,
  RowNumber INTEGER
);
```

The additional columns, AcademicYear, IngestedFileName, and RowNumber are basically derived from the metadata of the staged file.

We used the metadata columns **METADATA$FILENAME**, and **METADATA$FILE_ROW_NUMBER** for populating the new columns.

```
Please note that, for the files having header rows, column names will be in ROW_NUMBER 1
and data rows start from the second row.
```

The following Snowflake SQL will get us the filename and row number from the staged file.

```
SELECT METADATA$FILENAME, METADATA$FILE_ROW_NUMBER
FROM @IPEDS_HD;
```

Here is the mapping for additional columns in the case of academic institution staged files.

```
AcademicYear ⇒ SUBSTRING(METADATA$FILENAME,3,4)
IngestedFileName ⇒ METADATA$FILENAME
RowNumber  ⇒ METADATA$FILE_ROW_NUMBER
```

8.3 Snowflake view and stored procedure

We have introduced a Snowflake view and a stored procedure in between the staged files and operational data (OD) table to manage data processing using extract load and transfer (ELT). The view is built on top of the staged file to do necessary mapping and create appropriate column names for the result set. Then the stored procedure consumes this view and loads the OD table for a specific year.

8.4 Managing metadata of staged file

Now that, the academic institution information for years 2017, 2018, and 2019 is available in the staged file. The view **v_od_AcademicInstitution** is added using the mapping table defined earlier for the staged file @IPEDS_HD. Later, this view will be used for populating the staging table.

3 additional columns (AcademicYear, IngestedFileName, and RowNumber) required for the staging table are also defined in the view.

Here is the Snowflake command to create the view.

```
DROP VIEW IF EXISTS v_od_AcademicInstitution;

CREATE OR REPLACE VIEW v_od_AcademicInstitution  AS
SELECT
t.$1  AS InstitutionIdentifier,
t.$2  AS InstitutionName,
t.$3  AS InstitutionNameAlias,
t.$4  AS StreetAddress,
t.$5  AS City,
t.$6  AS State,
t.$7  AS ZipCode,
t.$8  AS StateCode,
t.$9  AS EconomicAnalysisRegions,
t.$10 AS ChiefAdministrator,
t.$11 AS ChiefAdministratorTitle,
t.$12 AS TelephoneNumber,
t.$13 AS EmployerIdentificationNumber,
t.$14 AS DunBradstreetNumbers,
t.$15 AS PostsecondaryEducationIDNumber,
t.$16 AS TitleIVEligibilityIndicatorCode,
t.$17 AS InstitutionsWebAddress,
t.$18 AS AdmissionsOfficeWebAddress,
t.$19 AS FinancialAidOfficeWebAddress,
t.$20 AS OnlineApplicationWebAddress,
t.$21 AS NetPriceCalculatorWebAddress,
t.$22 AS VeteransMilitaryServiceTuitionPoliciesWebAddress,
t.$23 AS StudentRightAthleteGraduationRateWebAddress,
t.$24 AS DisabilityServicesWebAddress,
t.$25 AS SectorOfInstitution,
t.$26 AS LevelOfInstitution,
t.$27 AS ControlOfInstitution,
t.$28 AS HighestLevelOfOffering,
t.$29 AS UndergraduateOffering,
t.$30 AS GraduateOffering,
t.$31 AS HighestDegreeOffered,
t.$32 AS DegreeGrantingStatus,
t.$33 AS HistoricallyBlackCollegeOrUniversity,
t.$34 AS InstitutionHasHospital,
t.$35 AS InstitutionGrantsMedicalDegree,
t.$36 AS TribalCollege,
t.$37 AS DegreeOfUrbanization,
t.$38 AS InstitutionOpenToGeneralPublic,
t.$39 AS StatusOfInstitution,
t.$40 AS UnitidForMergedSchools,
t.$41 AS YearInstitutionWasDeletedFromIPEDS,
t.$42 AS DateInstitutionClosed,
t.$43 AS InstitutionIsActive,
t.$44 AS PrimarilyPostsecondaryIndicator,
t.$45 AS PostsecondaryInstitutionIndicator,
t.$46 AS PostsecondaryAndTitleIvInstitutionIndicator,
t.$47 AS ReportingMethodForStudentCharges,
t.$48 AS InstitutionalCategory,
t.$49 AS CarnegieClassification2015Basic,
t.$50 AS CarnegieClassification2015UndergraduateProgram,
t.$51 AS CarnegieClassification2015GraduateProgram,
```

```
t.$52 AS CarnegieClassification2015UndergraduateProfile,
t.$53 AS CarnegieClassification2015EnrollmentProfile,
t.$54 AS CarnegieClassification2015SizeSetting,
t.$55 AS CarnegieClassification20052010Basic,
t.$56 AS CarnegieClassification2000,
t.$57 AS LandGrantInstitution,
t.$58 AS InstitutionSizeCategory,
t.$59 AS MultiCampusOrganization,
t.$60 AS NameOfMultiCampusOrganization,
t.$61 AS IdentificationNumberOfMultiCampusOrganization,
t.$62 AS CoreBasedStatisticalArea,
t.$63 AS CBSATypeMetropolitanMicropolitan,
t.$64 AS CombinedStatisticalArea,
t.$65 AS NewEnglandCityAndTownArea,
t.$66 AS FIPSCountyCode,
t.$67 AS CountyName,
t.$68 AS StateAnd114thCongressionalDistrictID,
t.$69 AS LongitudeLocation,
t.$70 AS LatitudeLocation,
t.$71 AS NCESGroupCategory,
t.$72 AS DataFeedbackReport,
CAST(substring(metadata$filename, 3, 4) AS INTEGER)
                        AcademicYear,
metadata$filename IngestedFileName,
metadata$file_row_number-1 RowNumber
FROM
   @IPEDS_HD t
WHERE
   metadata$file_row_number > 1;
```

** If the internal staged area has multiple files, the view created on staged will have access to all underlying files ingested.**

In Snowflake, views behave just like tables and we can retrieve the records for a given academic year using the view v_od_AcademicInstitution as below. As we ingested three files from the source system, @IPEDS_HD will have data from all academic years.

```
SELECT TOP 100 * FROM v_od_AcademicInstitution
WHERE AcademicYear = 2017;
```

8.5 Stored procedure for OD table load

Stored procedures are created to perform one or more DML operations. It is nothing but the group of SQL statements that accepts some input and performs some task and may or may not returns a value. In Snowflake, stored procedures are similar to functions. Stored procedures can be executed multiple times.

Stored procedures allow us to extend Snowflake SQL by combining it with JavaScript. It is created with a CREATE PROCEDURE command and is executed with a CALL command.

We've created a process to load the OD table od_AcademicInstitution data from the view, one year's data at a time. For this purpose, we have created a stored procedure **pr_od_AcademicInstitution_Load**. This stored procedure takes a parameter YEAR and retrieves the data from the view **v_od_AcademicInstitution** for that year. The retrieved records are then inserted into the OD table.

Here is the script for creating the Snowflake stored procedure pr_od_AcademicInstitution_Load.

```
CREATE OR REPLACE PROCEDURE
        pr_od_AcademicInstitution_Load (YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_command = `
   INSERT INTO   od_AcademicInstitution
    (InstitutionIdentifier,
    InstitutionName,
    InstitutionNameAlias,
    StreetAddress,
    City,
    State,
    ZipCode,
    StateCode,
    EconomicAnalysisRegions,
    ChiefAdministrator,
    ChiefAdministratorTitle,
    TelephoneNumber,
    EmployerIdentificationNumber,
    DunBradstreetNumbers,
    PostsecondaryEducationIDNumber,
    TitleIVEligibilityIndicatorCode,
    InstitutionsWebAddress,
    AdmissionsOfficeWebAddress,
    FinancialAidOfficeWebAddress,
    OnlineApplicationWebAddress,
    NetPriceCalculatorWebAddress,
    VeteransMilitaryServiceTuitionPoliciesWebAddress,
    StudentRightAthleteGraduationRateWebAddress,
    DisabilityServicesWebAddress,
    SectorOfInstitution,
    LevelOfInstitution,
    ControlOfInstitution,
    HighestLevelOfOffering,
    UndergraduateOffering,
    GraduateOffering,
    HighestDegreeOffered,
    DegreeGrantingStatus,
    HistoricallyBlackCollegeOrUniversity,
    InstitutionHasHospital,
    InstitutionGrantsMedicalDegree,
    TribalCollege,
    DegreeOfUrbanization,
    InstitutionOpenToGeneralPublic,
    StatusOfInstitution,
    UnitidForMergedSchools,
    YearInstitutionWasDeletedFromIPEDS,
    DateInstitutionClosed,
    InstitutionIsActive,
    PrimarilyPostsecondaryIndicator,
    PostsecondaryInstitutionIndicator,
    PostsecondaryAndTitleIvInstitutionIndicator,
```

```
    ReportingMethodForStudentCharges,
    InstitutionalCategory,
    CarnegieClassification2015UndergraduateProgram,
    CarnegieClassification2015GraduateProgram,
    CarnegieClassification2015UndergraduateProfile,
    CarnegieClassification2015EnrollmentProfile,
    CarnegieClassification2015SizeSetting,
    CarnegieClassification2015Basic,
    CarnegieClassification20052010Basic,
    CarnegieClassification2000,
    LandGrantInstitution,
    InstitutionSizeCategory,
    MultiCampusOrganization,
    NameOfMultiCampusOrganization,
    IdentificationNumberOfMultiCampusOrganization,
    CoreBasedStatisticalArea,
    CBSATypeMetropolitanMicropolitan,
    CombinedStatisticalArea,
    NewEnglandCityAndTownArea,
    FIPSCountyCode,
    CountyName,
    StateAnd114thCongressionalDistrictID,
    LongitudeLocation,
    LatitudeLocation,
    NCESGroupCategory,
    DataFeedbackReport,
    AcademicYear,
    IngestedFileName,
    RowNumber  )
        SELECT
          InstitutionIdentifier,
          InstitutionName,
          InstitutionNameAlias,
          StreetAddress,
          City,
          State,
          ZipCode,
          StateCode,
          EconomicAnalysisRegions,
          ChiefAdministrator,
          ChiefAdministratorTitle,
          TelephoneNumber,
          EmployerIdentificationNumber,
          DunBradstreetNumbers,
          PostsecondaryEducationIDNumber,
          TitleIVEligibilityIndicatorCode,
          InstitutionsWebAddress,
          AdmissionsOfficeWebAddress,
          FinancialAidOfficeWebAddress,
          OnlineApplicationWebAddress,
          NetPriceCalculatorWebAddress,
          VeteransMilitaryServiceTuitionPoliciesWebAddress,
          StudentRightAthleteGraduationRateWebAddress,
          DisabilityServicesWebAddress,
          SectorOfInstitution,
```

```
        LevelOfInstitution,
        ControlOfInstitution,
        HighestLevelOfOffering,
        UndergraduateOffering,
        GraduateOffering,
        HighestDegreeOffered,
        DegreeGrantingStatus,
        HistoricallyBlackCollegeOrUniversity,
        InstitutionHasHospital,
        InstitutionGrantsMedicalDegree,
        TribalCollege,
        DegreeOfUrbanization,
        InstitutionOpenToGeneralPublic,
        StatusOfInstitution,
        UnitidForMergedSchools,
        YearInstitutionWasDeletedFromIPEDS,
        DateInstitutionClosed,
        InstitutionIsActive,
        PrimarilyPostsecondaryIndicator,
        PostsecondaryInstitutionIndicator,
        PostsecondaryAndTitleIvInstitutionIndicator,
        ReportingMethodForStudentCharges,
        InstitutionalCategory,
        CarnegieClassification2015UndergraduateProgram,
        CarnegieClassification2015GraduateProgram,
        CarnegieClassification2015UndergraduateProfile,
        CarnegieClassification2015EnrollmentProfile,
        CarnegieClassification2015SizeSetting,
        CarnegieClassification2015Basic,
        CarnegieClassification20052010Basic,
        CarnegieClassification2000,
        LandGrantInstitution,
        InstitutionSizeCategory,
        MultiCampusOrganization,
        NameOfMultiCampusOrganization,
        IdentificationNumberOfMultiCampusOrganization,
        CoreBasedStatisticalArea,
        CBSATypeMetropolitanMicropolitan,
        CombinedStatisticalArea,
        NewEnglandCityAndTownArea,
        FIPSCountyCode,
        CountyName,
        StateAnd114thCongressionalDistrictID,
        LongitudeLocation,
        LatitudeLocation,
        NCESGroupCategory,
        DataFeedbackReport,
        AcademicYear,
        IngestedFileName,
        RowNumber
      FROM
        v_od_AcademicInstitution
  WHERE AcademicYear = ` + YEAR.toString()+`;`

  try {
```

```
        snowflake.execute (
          {sqlText: sql_command}
          );
        return "Succeeded.";  // Return a success/error indicator.
        }
    catch (err) {
        return "Failed: " + err;  // Return a success/error indicator.
        }
$$
;
```

Orchestration engine needs to delete old data from OD table and load data.

The OD table is truncated and data for 2017 is loaded as shown below.

```
TRUNCATE TABLE od_AcademicInstitution;
CALL pr_od_AcademicInstitution_Load(2017::FLOAT);
```

Now, the OD table is filled with Academic Institution data for year 2017.

```
SELECT TOP 50 * FROM od_AcademicInstitution;
```

Here is the sample data retrieved using the SQL given above.

InstitutionIdentifier	InstitutionName	InstitutionNameAlias	StreetAddress	City	State	Zipcode	.....	DataFeedbackReport	AcademicYear	IngestedFileName	RowNumber
100654	Alabama A & M University	AAMU	4900 Meridian Street	Normal	AL	35762		1	2017	HD2017.csv.gz	1
100663	University of Alabama at Birmingham		Administration Bldg Suite 1070	Birmingham	AL	35294-0110		1	2017	HD2017.csv.gz	2
100690	Amridge University	Southern Christian University |Regions University	1200 Taylor Rd	Montgomery	AL	36117-3553		2	2017	HD2017.csv.gz	3
100706	University of Alabama in Huntsville	UAH |University of Alabama Huntsville	301 Sparkman Dr	Huntsville	AL	35899		2	2017	HD2017.csv.gz	4
100724	Alabama State University		915 S Jackson Street	Montgomery	AL	36104-0271		1	2017	HD2017.csv.gz	5
100733	University of Alabama System Office		500 University Blvd. East	Tuscaloosa	AL	35401		-2	2017	HD2017.csv.gz	6
100751	The University of Alabama		739 University Blvd	Tuscaloosa	AL	35487-0166		1	2017	HD2017.csv.gz	7
100760	Central Alabama Community College		1675 Cherokee Rd	Alexander City	AL	35010		2	2017	HD2017.csv.gz	8
100812	Athens State University		300 N Beaty St	Athens	AL	35611		1	2017	HD2017.csv.gz	9
100830	Auburn University at Montgomery	AUM|Auburn University Montgomery|Auburn University at Montgomery|Auburn Montgomery	7440 East Drive	Montgomery	AL	36117-3596		1	2017	HD2017.csv.gz	10
100858	Auburn University			Auburn	AL	36849		1	2017	HD2017.csv.gz	11
100937	Birmingham Southern College		900 Arkadelphia Road	Birmingham	AL	35254		1	2017	HD2017.csv.gz	12
101028	Chattahoochee Valley Community College	CVCC	2602 College Drive	Phenix City	AL	36869		2	2017	HD2017.csv.gz	13
101077	Concordia College Alabama	CCA	1712 Broad Street	Selma	AL	36701		2	2017	HD2017.csv.gz	14
101116	South University-Montgomery		5355 Vaughn Rd	Montgomery	AL	36116		1	2017	HD2017.csv.gz	15
101143	Enterprise State Community College	Enterprise-Ozark Community College |Enterprise State Junior College	600 Plaza Drive	Enterprise	AL	36330-1300		2	2017	HD2017.csv.gz	16
101161	Coastal Alabama Community College		1900 US Hwy 31 S	Bay Minette	AL	36507-2698		1	2017	HD2017.csv.gz	17
101189	Faulkner University	Faulkner	5345 Atlanta Hwy	Montgomery	AL	36109-3390		2	2017	HD2017.csv.gz	18
101240	Gadsden State Community College	GSCC | Gadsden State | Gadsden Community College	1001 George Wallace Dr	Gadsden	AL	35903		2	2017	HD2017.csv.gz	19
101277	New Beginning College of Cosmetology		421 Martling Road	Albertville	AL	35951		1	2017	HD2017.csv.gz	20
101286	George C Wallace Community College-Dothan	WALLACE DOTHAN|WCCD	1141 Wallace Drive	Dothan	AL	36303-9234		2	2017	HD2017.csv.gz	21
101295	George C Wallace State Community College-Hanceville	Wallace State Community College |Wallace State - Hanceville	801 Main St NW	Hanceville	AL	35077-2000		1	2017	HD2017.csv.gz	22
101301	George C Wallace State Community College-Selma	Wallace Community College - Selma	3000 Earl Goodwin Parkway	Selma	AL	36703-2808		2	2017	HD2017.csv.gz	23
101365	Herzing University-Birmingham		280 West Valley Ave	Birmingham	AL	35209		1	2017	HD2017.csv.gz	24

8.6 Academic Institution data processing

Using the Academic Institution files, we have built a dimension table called **AcademicInstitution**. This dimension table will be used by fact tables at the servicing layer in forthcoming chapters.

In general, a dimension table contains keys columns and other attributes which are associated with the business key. A system-generated key (primary key) is used to uniquely identify a row in the dimension. This key is also known as a surrogate key. In a dimensional data warehouse architecture, the surrogate key is placed in the fact table and a foreign key is defined to establish the relationship between the two tables.

In the case of the dimension table AcademicInstitution, we have defined the column **AcademicInstitutionUniqueDWSID** as the primary key. This primary key gets a unique sequence value for every record inserted. The automatic sequencing is achieved in Snowflake by defining **SEQUENCE** and **NEXTVAL**.

Here is the script for creating the sequence SEQ_IPEDS.

```
CREATE OR REPLACE SEQUENCE
        SEQ_IPEDS_HD START = 1 INCREMENT = 1;
```

Following Snowflake SQL script is used for creating the AcademicInstitution dimension.

```
DROP TABLE IF EXISTS AcademicInstitution;

CREATE TABLE AcademicInstitution (
  AcademicInstitutionUniqueDWSID INTEGER
               NOT NULL DEFAULT SEQ_IPEDS_HD.NEXTVAL,
  InstitutionIdentifier INTEGER,
  InstitutionName VARCHAR(2000),
  InstitutionNameAlias VARCHAR(2000),
  StreetAddress VARCHAR(500),
  City VARCHAR(100),
  State VARCHAR(20),
  ZipCode VARCHAR(15),
  StateCode INTEGER,
  EconomicAnalysisRegions INTEGER,
  ChiefAdministrator VARCHAR(500),
  ChiefAdministratorTitle VARCHAR(500),
  TelephoneNumber VARCHAR(15),
  EmployerIdentificationNumber VARCHAR(50),
  DunBradstreetNumbers VARCHAR(2000),
  PostsecondaryEducationIDNumber VARCHAR(8),
  TitleIVEligibilityIndicatorCode INTEGER,
  SectorOfInstitution INTEGER,
  LevelOfInstitution INTEGER,
  ControlOfInstitution INTEGER,
  HighestLevelOfOffering INTEGER,
  UndergraduateOffering INTEGER,
  GraduateOffering INTEGER,
  HighestDegreeOffered INTEGER,
  DegreeGrantingStatus INTEGER,
  HistoricallyBlackCollegeOrUniversity INTEGER,
  InstitutionHasHospital INTEGER,
  InstitutionGrantsMedicalDegree INTEGER,
  TribalCollege INTEGER,
  DegreeOfUrbanization INTEGER,
  InstitutionOpenToGeneralPublic INTEGER,
  StatusOfInstitution VARCHAR(100),
  UnitidForMergedSchools VARCHAR(2000),
  YearInstitutionWasDeletedFromIPEDS INTEGER,
  DateInstitutionClosed VARCHAR(100),
  InstitutionIsActive INTEGER,
  PrimarilyPostsecondaryIndicator INTEGER,
  PostsecondaryInstitutionIndicator INTEGER,
  PostsecondaryAndTitleIvInstitutionIndicator INTEGER,
  ReportingMethodForStudentCharges INTEGER,
  InstitutionalCategory INTEGER,
  LandGrantInstitution INTEGER,
  InstitutionSizeCategory INTEGER,
  MultiCampusOrganization INTEGER,
  NameOfMultiCampusOrganization VARCHAR(800),
```

```
  IdentificationNumberOfMultiCampusOrganization VARCHAR(60),
  CoreBasedStatisticalArea INTEGER,
  CBSATypeMetropolitanMicropolitan INTEGER,
  CombinedStatisticalArea INTEGER,
  NewEnglandCityAndTownArea INTEGER,
  FIPSCountyCode INTEGER,
  CountyName VARCHAR(300),
  StateAnd114thCongressionalDistrictID INTEGER,
  LongitudeLocation NUMBER(32, 0),
  LatitudeLocation NUMBER(32, 0),
  NCESGroupCategory INTEGER,
  CarnegieClassification VARIANT,
  WebAddress VARIANT,
  DataFeedbackReport INTEGER,
  AcademicYear  INTEGER,
  RecordCreateDateTime DATETIME,
  RecordUpdateDateTime DATETIME
);
```

Note

The VARIANT columns (CarnegieClassification, WebAddress) are added to the dimension table. These columns are used to store JSON data in the table. The DATETIME columns RecordCreateDateTime and RecordUpdateDateTime are added for auditing data Upserts.

8.6.1 Carnegie Classification transformation

We have grouped together certain columns from the OD table. The following columns belong to CarnegieClassification and are stored in the dimension table in JSON format. Hence, the column CarnegieClassification is created with the VARIANT data type.

- CarnegieClassification2000
- CarnegieClassification20052010Basic
- CarnegieClassification2015Basic
- CarnegieClassification2015UndergraduateProgram
- CarnegieClassification2015GraduateProgram
- CarnegieClassification2015UndergraduateProfile
- CarnegieClassification2015EnrollmentProfile
- CarnegieClassification2015SizeSetting

8.6.2 Web Address transformation

Similarly, we have grouped together below columns holding various web addresses under the name WebAddress and stored in a JSON format. Thus another VARIANT data type column WebAddress is also created.

- InstitutionsWebAddress

- AdmissionsOfficeWebAddress
- FinancialAidOfficeWebAddress
- OnlineApplicationWebAddress
- NetPriceCalculatorWebAddress
- VeteransMilitaryServiceTuitionPoliciesWebAddress
- StudentRightAthleteGraduationRateWebAddress
- DisabilityServicesWebAddress

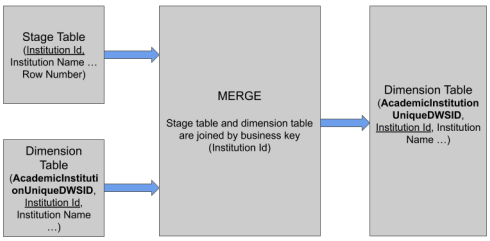
8.7 Load Dimension table

Once the dimension table is created, we need to load the dimension table **AcademicInstitution** with data from the operational data table **od_AcademicInstitution**. This is made possible by using the Snowflake MERGE command.

The MERGE command uses source and target tables for the merge operation. In our case, the OD table **od_AcademicInstitution** is the source table and the dimension table **AcademicInstitution** is the target table. The business key is required for MERGE and for dimension **AcademicInstitution** ' business key is **InstitutionIdentifier**.

In Snowflake MERGE works similar to standard SQL MERGE. We join the source and target using the business key. Then we compare the new data in the OD table against the existing column values in the dimension table. If any of the column values are updated in the staging table, that value is updated in the dimension table. If the business key **InstitutionIdentifier** is not present in the dimension table (new records in the staging table), that record is inserted into the dimension table.

We are using the slowly changing Type 1 concepts from datawarehouse load for merging AcademicInstitution.



We have created a stored procedure **pr_AcademicInstitution_Load** for merging data from the staging table **od_AcademicInstitution** to the dimension table **AcademicInstitution**. The script to create the stored procedure is given below.

```
CREATE OR REPLACE PROCEDURE
        pr_AcademicInstitution_Load(YEAR FLOAT)
  RETURNS STRING
  LANGUAGE javascript
  EXECUTE AS OWNER
  AS
  $$
var sql_command =
`
MERGE INTO AcademicInstitution T USING (
```

```
SELECT  * FROM  od_AcademicInstitution
WHERE  ACADEMICYEAR = ` + YEAR.toString() +`
) S ON T.InstitutionIdentifier = S.InstitutionIdentifier
WHEN MATCHED
AND (
  IFNULL(T.InstitutionName, '') <> IFNULL(S.InstitutionName, '')
  OR IFNULL(T.InstitutionNameAlias, '')
        <> IFNULL(S.InstitutionNameAlias, '')
  OR IFNULL(T.StreetAddress, '') <> IFNULL(S.StreetAddress, '')
  OR IFNULL(T.City, '') <> IFNULL(S.City, '')
  OR IFNULL(T.State, '') <> IFNULL(S.State, '')
  OR IFNULL(T.ZipCode, '') <> IFNULL(S.ZipCode, '')
  OR IFNULL(T.StateCode, 0) <> IFNULL(S.StateCode, 0)
  OR IFNULL(T.EconomicAnalysisRegions, 0)
        <> IFNULL(S.EconomicAnalysisRegions, 0)
  OR IFNULL(T.ChiefAdministrator, '')
        <> IFNULL(S.ChiefAdministrator, '')
  OR IFNULL(T.ChiefAdministratorTitle, '')
        <> IFNULL(S.ChiefAdministratorTitle, '')
  OR IFNULL(T.TelephoneNumber, '') <> IFNULL(S.TelephoneNumber, '')
  OR IFNULL(T.EmployerIdentificationNumber, '')
        <> IFNULL(S.EmployerIdentificationNumber, '')
  OR IFNULL(T.DunBradstreetNumbers, '')
        <> IFNULL(S.DunBradstreetNumbers, '')
  OR IFNULL(T.PostsecondaryEducationIDNumber, '')
        <> IFNULL(S.PostsecondaryEducationIDNumber, '')
  OR IFNULL(T.TitleIVEligibilityIndicatorCode, 0)
        <> IFNULL(S.TitleIVEligibilityIndicatorCode, 0)
  OR IFNULL(T.SectorOfInstitution, 0)
        <> IFNULL(S.SectorOfInstitution, 0)
  OR IFNULL(T.LevelOfInstitution, 0)
        <> IFNULL(S.LevelOfInstitution, 0)
  OR IFNULL(T.ControlOfInstitution, 0)
        <> IFNULL(S.ControlOfInstitution, 0)
  OR IFNULL(T.HighestLevelOfOffering, 0)
        <> IFNULL(S.HighestLevelOfOffering, 0)
  OR IFNULL(T.UndergraduateOffering, 0)
        <> IFNULL(S.UndergraduateOffering, 0)
  OR IFNULL(T.GraduateOffering, 0)
        <> IFNULL(S.GraduateOffering, 0)
  OR IFNULL(T.HighestDegreeOffered, 0)
        <> IFNULL(S.HighestDegreeOffered, 0)
  OR IFNULL(T.DegreeGrantingStatus, 0)
        <> IFNULL(S.DegreeGrantingStatus, 0)
  OR IFNULL(T.HistoricallyBlackCollegeOrUniversity, 0)
        <> IFNULL(S.HistoricallyBlackCollegeOrUniversity, 0)
  OR IFNULL(T.InstitutionHasHospital, 0)
        <> IFNULL(S.InstitutionHasHospital, 0)
  OR IFNULL(T.InstitutionGrantsMedicalDegree, 0)
        <> IFNULL(S.InstitutionGrantsMedicalDegree, 0)
  OR IFNULL(T.TribalCollege, 0) <> IFNULL(S.TribalCollege, 0)
  OR IFNULL(T.DegreeOfUrbanization, 0)
        <> IFNULL(S.DegreeOfUrbanization, 0)
  OR IFNULL(T.InstitutionOpenToGeneralPublic, 0)
        <> IFNULL(S.InstitutionOpenToGeneralPublic, 0)
```

```
 OR IFNULL(T.StatusOfInstitution, '')
       <> IFNULL(S.StatusOfInstitution, '')
 OR IFNULL(T.UnitidForMergedSchools, 0)
       <> IFNULL(S.UnitidForMergedSchools, 0)
 OR IFNULL(T.YearInstitutionWasDeletedFromIPEDS, 0)
       <> IFNULL(S.YearInstitutionWasDeletedFromIPEDS, 0)
 OR IFNULL(T.DateInstitutionClosed, '')
       <> IFNULL(S.DateInstitutionClosed, '')
 OR IFNULL(T.InstitutionIsActive, 0)
       <> IFNULL(S.InstitutionIsActive, 0)
 OR IFNULL(T.PrimarilyPostsecondaryIndicator, 0)
       <> IFNULL(S.PrimarilyPostsecondaryIndicator, 0)
 OR IFNULL(T.PostsecondaryInstitutionIndicator, 0)
       <> IFNULL(S.PostsecondaryInstitutionIndicator, 0)
 OR IFNULL(T.PostsecondaryAndTitleIvInstitutionIndicator, 0)
       <> IFNULL(S.PostsecondaryAndTitleIvInstitutionIndicator, 0)
 OR IFNULL(T.ReportingMethodForStudentCharges, 0)
       <> IFNULL(S.ReportingMethodForStudentCharges, 0)
 OR IFNULL(T.InstitutionalCategory, 0)
       <> IFNULL(S.InstitutionalCategory, 0)
 OR IFNULL(T.LandGrantInstitution, 0)
       <> IFNULL(S.LandGrantInstitution, 0)
 OR IFNULL(T.InstitutionSizeCategory, 0)
       <> IFNULL(S.InstitutionSizeCategory, 0)
 OR IFNULL(T.MultiCampusOrganization, 0)
       <> IFNULL(S.MultiCampusOrganization, 0)
 OR IFNULL(T.NameOfMultiCampusOrganization, '')
       <> IFNULL(S.NameOfMultiCampusOrganization, '')
 OR IFNULL(T.IdentificationNumberOfMultiCampusOrganization, '')
       <> IFNULL(S.IdentificationNumberOfMultiCampusOrganization,'')
 OR IFNULL(T.CoreBasedStatisticalArea, 0)
       <> IFNULL(S.CoreBasedStatisticalArea, 0)
 OR IFNULL(T.CBSATypeMetropolitanMicropolitan, 0)
       <> IFNULL(S.CBSATypeMetropolitanMicropolitan, 0)
 OR IFNULL(T.CombinedStatisticalArea, 0)
       <> IFNULL(S.CombinedStatisticalArea, 0)
 OR IFNULL(T.NewEnglandCityAndTownArea, 0)
       <> IFNULL(S.NewEnglandCityAndTownArea, 0)
 OR IFNULL(T.FIPSCountyCode, 0) <> IFNULL(S.FIPSCountyCode, 0)
 OR IFNULL(T.CountyName, '') <> IFNULL(S.CountyName, '')
 OR IFNULL(T.StateAnd114thCongressionalDistrictID, 0)
       <> IFNULL(S.StateAnd114thCongressionalDistrictID, 0)
 OR IFNULL(T.LongitudeLocation, 0.0)
       <> IFNULL(S.LongitudeLocation, 0.0)
 OR IFNULL(T.LatitudeLocation, 0.0)
       <> IFNULL(S.LatitudeLocation, 0.0)
 OR IFNULL(T.NCESGroupCategory, 0)
       <> IFNULL(S.NCESGroupCategory, 0)
 OR IFNULL(T.DataFeedbackReport, 0)
       <> IFNULL(S.DataFeedbackReport, 0)
 OR CarnegieClassification  <> OBJECT_CONSTRUCT (
       'AcademicInstitutionIdentifier'
              ,S.InstitutionIdentifier,
       'CarnegieClassification2000'
              ,S.CarnegieClassification2000,
```

```
        'CarnegieClassification20052010Basic'
                ,S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic'
                ,S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram'
                ,S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram'
                ,S.CarnegieClassification2015GraduateProgram,
        'CarnegieClassification2015UndergraduateProfile'
                ,S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile'
                ,S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting'
                ,S.CarnegieClassification2015SizeSetting
                        )
  OR WebAddress <> OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier'
                ,S.InstitutionIdentifier,
        'InstitutionsWebAddress'
                ,S.InstitutionsWebAddress,
        'AdmissionsOfficeWebAddress'
                ,S.AdmissionsOfficeWebAddress,
        'FinancialAidOfficeWebAddress'
                ,S.FinancialAidOfficeWebAddress,
        'OnlineApplicationWebAddress'
                ,S.OnlineApplicationWebAddress,
        'NetPriceCalculatorWebAddress'
                ,S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress'
                ,S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress'
                ,S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress'
                ,S.DisabilityServicesWebAddress
                   )

) THEN
UPDATE
SET
  InstitutionIdentifier = S.InstitutionIdentifier,
  InstitutionName = S.InstitutionName,
  InstitutionNameAlias = S.InstitutionNameAlias,
  StreetAddress = S.StreetAddress,
  City = S.City,
  State = S.State,
  ZipCode = S.ZipCode,
  StateCode = S.StateCode,
  EconomicAnalysisRegions = S.EconomicAnalysisRegions,
  ChiefAdministrator = S.ChiefAdministrator,
  ChiefAdministratorTitle = S.ChiefAdministratorTitle,
  TelephoneNumber = S.TelephoneNumber,
  EmployerIdentificationNumber =
                S.EmployerIdentificationNumber,
  DunBradstreetNumbers = S.DunBradstreetNumbers,
  PostsecondaryEducationIDNumber =
```

```
              S.PostsecondaryEducationIDNumber,
TitleIVEligibilityIndicatorCode =
              S.TitleIVEligibilityIndicatorCode,
SectorOfInstitution = S.SectorOfInstitution,
LevelOfInstitution = S.LevelOfInstitution,
ControlOfInstitution = S.ControlOfInstitution,
HighestLevelOfOffering = S.HighestLevelOfOffering,
UndergraduateOffering = S.UndergraduateOffering,
GraduateOffering = S.GraduateOffering,
HighestDegreeOffered = S.HighestDegreeOffered,
DegreeGrantingStatus = S.DegreeGrantingStatus,
HistoricallyBlackCollegeOrUniversity
              = S.HistoricallyBlackCollegeOrUniversity,
InstitutionHasHospital =
              S.InstitutionHasHospital,
InstitutionGrantsMedicalDegree =
              S.InstitutionGrantsMedicalDegree,
TribalCollege = S.TribalCollege,
DegreeOfUrbanization = S.DegreeOfUrbanization,
InstitutionOpenToGeneralPublic =
              S.InstitutionOpenToGeneralPublic,
StatusOfInstitution = S.StatusOfInstitution,
UnitidForMergedSchools = S.UnitidForMergedSchools,
YearInstitutionWasDeletedFromIPEDS
              = S.YearInstitutionWasDeletedFromIPEDS,
DateInstitutionClosed = S.DateInstitutionClosed,
InstitutionIsActive = S.InstitutionIsActive,
PrimarilyPostsecondaryIndicator =
              S.PrimarilyPostsecondaryIndicator,
PostsecondaryInstitutionIndicator =
              S.PostsecondaryInstitutionIndicator,
PostsecondaryAndTitleIvInstitutionIndicator
              = S.PostsecondaryAndTitleIvInstitutionIndicator,
ReportingMethodForStudentCharges
              = S.ReportingMethodForStudentCharges,
InstitutionalCategory = S.InstitutionalCategory,
LandGrantInstitution = S.LandGrantInstitution,
InstitutionSizeCategory = S.InstitutionSizeCategory,
MultiCampusOrganization = S.MultiCampusOrganization,
NameOfMultiCampusOrganization =
              S.NameOfMultiCampusOrganization,
IdentificationNumberOfMultiCampusOrganization
              = S.IdentificationNumberOfMultiCampusOrganization,
CoreBasedStatisticalArea =
              S.CoreBasedStatisticalArea,
CBSATypeMetropolitanMicropolitan =
              S.CBSATypeMetropolitanMicropolitan,
CombinedStatisticalArea = S.CombinedStatisticalArea,
NewEnglandCityAndTownArea = S.NewEnglandCityAndTownArea,
FIPSCountyCode = S.FIPSCountyCode,
CountyName = S.CountyName,
StateAnd114thCongressionalDistrictID
              = S.StateAnd114thCongressionalDistrictID,
LongitudeLocation = S.LongitudeLocation,
LatitudeLocation = S.LatitudeLocation,
```

```
  NCESGroupCategory = S.NCESGroupCategory,
  DataFeedbackReport = S.DataFeedbackReport,
  CarnegieClassification =OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier'
                ,S.InstitutionIdentifier,
        'CarnegieClassification2000'
                ,S.CarnegieClassification2000,
        'CarnegieClassification20052010Basic'
                ,S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic'
                ,S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram'
                ,S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram'
                ,S.CarnegieClassification2015GraduateProgram,
        'CarnegieClassification2015UndergraduateProfile'
                ,S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile'
                ,S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting'
                ,S.CarnegieClassification2015SizeSetting
          ),
  WebAddress=OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier'
                ,S.InstitutionIdentifier,
        'InstitutionsWebAddress'
                ,S.InstitutionsWebAddress,
        'AdmissionsOfficeWebAddress'
                ,S.AdmissionsOfficeWebAddress,
        'FinancialAidOfficeWebAddress'
                ,S.FinancialAidOfficeWebAddress,
        'OnlineApplicationWebAddress'
                ,S.OnlineApplicationWebAddress,
        'NetPriceCalculatorWebAddress'
                ,S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress'
                ,S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress'
                ,S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress'
                ,S.DisabilityServicesWebAddress
          ),
  AcademicYear=S.AcademicYear,
  RecordUpdateDateTime = CURRENT_TIMESTAMP
  WHEN NOT MATCHED THEN
INSERT
  (
        InstitutionIdentifier,
        InstitutionName,
        InstitutionNameAlias,
        StreetAddress,
        City,
        State,
        ZipCode,
        StateCode,
```

```
        EconomicAnalysisRegions,
        ChiefAdministrator,
        ChiefAdministratorTitle,
        TelephoneNumber,
        EmployerIdentificationNumber,
        DunBradstreetNumbers,
        PostsecondaryEducationIDNumber,
        TitleIVEligibilityIndicatorCode,
        SectorOfInstitution,
        LevelOfInstitution,
        ControlOfInstitution,
        HighestLevelOfOffering,
        UndergraduateOffering,
        GraduateOffering,
        HighestDegreeOffered,
        DegreeGrantingStatus,
        HistoricallyBlackCollegeOrUniversity,
        InstitutionHasHospital,
        InstitutionGrantsMedicalDegree,
        TribalCollege,
        DegreeOfUrbanization,
        InstitutionOpenToGeneralPublic,
        StatusOfInstitution,
        UnitidForMergedSchools,
        YearInstitutionWasDeletedFromIPEDS,
        DateInstitutionClosed,
        InstitutionIsActive,
        PrimarilyPostsecondaryIndicator,
        PostsecondaryInstitutionIndicator,
        PostsecondaryAndTitleIvInstitutionIndicator,
        ReportingMethodForStudentCharges,
        InstitutionalCategory,
        LandGrantInstitution,
        InstitutionSizeCategory,
        MultiCampusOrganization,
        NameOfMultiCampusOrganization,
        IdentificationNumberOfMultiCampusOrganization,
        CoreBasedStatisticalArea,
        CBSATypeMetropolitanMicropolitan,
        CombinedStatisticalArea,
        NewEnglandCityAndTownArea,
        FIPSCountyCode,
        CountyName,
        StateAnd114thCongressionalDistrictID,
        LongitudeLocation,
        LatitudeLocation,
        NCESGroupCategory,
        DataFeedbackReport,
        AcademicYear,
        CarnegieClassification,
        WebAddress,
        RecordCreateDateTime)
VALUES
  (
        S.InstitutionIdentifier,
```

```
        S.InstitutionName,
        S.InstitutionNameAlias,
        S.StreetAddress,
        S.City,
        S.State,
        S.ZipCode,
        S.StateCode,
        S.EconomicAnalysisRegions,
        S.ChiefAdministrator,
        S.ChiefAdministratorTitle,
        S.TelephoneNumber,
        S.EmployerIdentificationNumber,
        S.DunBradstreetNumbers,
        S.PostsecondaryEducationIDNumber,
        S.TitleIVEligibilityIndicatorCode,
        S.SectorOfInstitution,
        S.LevelOfInstitution,
        S.ControlOfInstitution,
        S.HighestLevelOfOffering,
        S.UndergraduateOffering,
        S.GraduateOffering,
        S.HighestDegreeOffered,
        S.DegreeGrantingStatus,
        S.HistoricallyBlackCollegeOrUniversity,
        S.InstitutionHasHospital,
        S.InstitutionGrantsMedicalDegree,
        S.TribalCollege,
        S.DegreeOfUrbanization,
        S.InstitutionOpenToGeneralPublic,
        S.StatusOfInstitution,
        S.UnitidForMergedSchools,
        S.YearInstitutionWasDeletedFromIPEDS,
        S.DateInstitutionClosed,
        S.InstitutionIsActive,
        S.PrimarilyPostsecondaryIndicator,
        S.PostsecondaryInstitutionIndicator,
        S.PostsecondaryAndTitleIvInstitutionIndicator,
        S.ReportingMethodForStudentCharges,
        S.InstitutionalCategory,
        S.LandGrantInstitution,
        S.InstitutionSizeCategory,
        S.MultiCampusOrganization,
        S.NameOfMultiCampusOrganization,
        S.IdentificationNumberOfMultiCampusOrganization,
        S.CoreBasedStatisticalArea,
        S.CBSATypeMetropolitanMicropolitan,
        S.CombinedStatisticalArea,
        S.NewEnglandCityAndTownArea,
        S.FIPSCountyCode,
        S.CountyName,
        S.StateAnd114thCongressionalDistrictID,
        S.LongitudeLocation,
        S.LatitudeLocation,
        S.NCESGroupCategory,
        S.DataFeedbackReport,
```

```
        S.AcademicYear        ,
        OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier'
                ,S.InstitutionIdentifier,
        'CarnegieClassification2000'
                ,S.CarnegieClassification2000,
        'CarnegieClassification20052010Basic'
                ,S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic'
                ,S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram'
                ,S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram'
                ,S.CarnegieClassification2015GraduateProgram,
        'CarnegieClassification2015UndergraduateProfile'
                ,S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile'
                ,S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting'
                ,S.CarnegieClassification2015SizeSetting
           ),
        OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier'
                ,S.InstitutionIdentifier,
        'InstitutionsWebAddress'
                ,S.InstitutionsWebAddress,
        'AdmissionsOfficeWebAddress'
                ,S.AdmissionsOfficeWebAddress,
        'FinancialAidOfficeWebAddress'
                ,S.FinancialAidOfficeWebAddress,
        'OnlineApplicationWebAddress'
                ,S.OnlineApplicationWebAddress,
        'NetPriceCalculatorWebAddress'
                ,S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress'
                ,S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress'
                ,S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress'
                ,S.DisabilityServicesWebAddress
                       ),
                        CURRENT_TIMESTAMP
                  );
        `

try {
    snowflake.execute (
      {sqlText: sql_command}
      );
    return "Succeeded.";  // Return a success/error indicator.
    }
  catch (err) {
    return "Failed: " + err;  // Return a success/error indicator.
    }
  $$
  ;
```

And the stored procedure is called to load AcademicInstitution table from OD.

```
CALL pr_AcademicInstitution_Load(2017::FLOAT) ;
-- Check data on underlying table
SELECT * FROM AcademicInstitution
```

The following diagram shows all the data being processed and AcademicInstitution has The data as desired

Row	INSTITUTIONID	INSTITUTIONNA	INSTITUTIONNA	STREETADDRES	CITY	STATE	ZIPCODE	STATECODE	ECONOMICANA	CHIEFADMINIST	CHIEFADMINIS
1	100654	Alabama A &...	AAMU	4900 Meridi...	Normal	AL	35762	1	5	Dr. Andrew ...	President
2	100663	University of...		Administrati...	Birmingham	AL	35294-0110	1	5	Ray L. Watts	President
3	100690	Amridge Uni...	Southern Ch...	1200 Taylor ...	Montgomery	AL	36117-3553	1	5	Michael C.Tu...	President
4	100706	University of...	UAH |Univer...	301 Sparkm...	Huntsville	AL	35899	1	5	Robert A. Alt...	President
5	100724	Alabama Sta...		915 S Jacks...	Montgomery	AL	36104-0271	1	5	Quinton T. R...	President
6	100733	University of...		500 Universi...	Tuscaloosa	AL	35401	1	5	Ray Hayes	Chancellor
7	100751	The Universi...		739 Universi...	Tuscaloosa	AL	35487-0166	1	5	Dr. Stuart R. ...	President
8	100760	Central Alab...		1675 Cherok...	Alexander City	AL	35010	1	5	Dr. Susan Bu...	President
9	100812	Athens State...		300 N Beaty...	Athens	AL	35611	1	5	Robert Glenn	President
10	100830	Auburn Univ...	AUM|Auburn...	7440 East D...	Montgomery	AL	36117-3596	1	5	Carl A Stock...	Chancellor
11	100858	Auburn Univ...			Auburn	AL	36849	1	5	Steven Leath	President
12	100937	Birmingham ...		900 Arkadel...	Birmingham	AL	35254	1	5	Ms. Linda Fl...	President
13	101028	Chattahooc...	CVCC	2602 Colleg...	Phenix City	AL	36869	1	5	Jacqueline S...	President

8.8 Schema drift

Structural changes in the incoming data files are very common in a data warehouse environment. In our case from 2018, we have noticed the following structural changes in the input files generated from source system.

The following new columns are added to files from 2018 onwards.

- CarnegieClassification2018Basic
- CarnegieClassification2018UndergraduateProgram
- CarnegieClassification2018GraduateProgram
- CarnegieClassification2018UndergraduateProfile
- CarnegieClassification2018EnrollmentProfile
- CarnegieClassification2018SizeSetting

And the following columns are deleted from the academic institution files from 2018 onwards.

- CarnegieClassification2015UndergraduateProgram
- CarnegieClassification2015GraduateProgram
- CarnegieClassification2015UndergraduateProfile
- CarnegieClassification2015EnrollmentProfile
- CarnegieClassification2015SizeSetting

Also, there are some changes in position of a few existing columns in the file.

Our goal is to process both old and new files using the same load process to load **od_AcademicInstitution**. We have added the new columns to the OD table using ALTER TABLE as below.

```
ALTER TABLE  od_AcademicInstitution
ADD (
CarnegieClassification2018Basic INTEGER,
CarnegieClassification2018UndergraduateProgram INTEGER,
CarnegieClassification2018GraduateProgram INTEGER,
CarnegieClassification2018UndergraduateProfile INTEGER,
CarnegieClassification2018EnrollmentProfile INTEGER,
CarnegieClassification2018SizeSetting INTEGER);
```

Also, we updated the view **v_od_AcademicInstitution** to include newly added columns using the Snowflake UNION command as below. This has made it possible to ingest files with different structures for 2017, and more recent years.

The deleted columns are populated with NULL values for the year greater than 2017. Newly added columns are mapped with NULL for the year 2017.

We have to modify the following Snowflake data objects to incorporate the above structural change along with table change in the tables **od_AcademicInstitution** and **AcademicInstitution** .

1. View : v_od_AcademicInstitution
2. Stored procedure for loading staging table : pr_od_AcademicInstitution_Load
3. Stored procedure for processing dimension table : pr_AcademicInstitution_Load

Modifying the view from the staged file

```
DROP VIEW IF EXISTS v_od_AcademicInstitution;

CREATE OR REPLACE VIEW v_od_AcademicInstitution  AS
SELECT
  t.$1 AS InstitutionIdentifier,
  t.$2 AS InstitutionName,
  t.$3 AS InstitutionNameAlias,
  t.$4 AS StreetAddress,
  t.$5 AS City,
  t.$6 AS State,
  t.$7 AS ZipCode,
  t.$8 AS StateCode,
  t.$9 AS EconomicAnalysisRegions,
  t.$10 AS ChiefAdministrator,
  t.$11 AS ChiefAdministratorTitle,
  t.$12 AS TelephoneNumber,
  t.$13 AS EmployerIdentificationNumber,
  t.$14 AS DunBradstreetNumbers,
  t.$15 AS PostsecondaryEducationIDNumber,
  t.$16 AS TitleIVEligibilityIndicatorCode,
  t.$17 AS InstitutionsWebAddress,
  t.$18 AS AdmissionsOfficeWebAddress,
  t.$19 AS FinancialAidOfficeWebAddress,
  t.$20 AS OnlineApplicationWebAddress,
  t.$21 AS NetPriceCalculatorWebAddress,
  t.$22 AS VeteransMilitaryServiceTuitionPoliciesWebAddress,
  t.$23 AS StudentRightAthleteGraduationRateWebAddress,
  t.$24 AS DisabilityServicesWebAddress,
  t.$25 AS SectorOfInstitution,
  t.$26 AS LevelOfInstitution,
```

```
  t.$27 AS ControlOfInstitution,
  t.$28 AS HighestLevelOfOffering,
  t.$29 AS UndergraduateOffering,
  t.$30 AS GraduateOffering,
  t.$31 AS HighestDegreeOffered,
  t.$32 AS DegreeGrantingStatus,
  t.$33 AS HistoricallyBlackCollegeOrUniversity,
  t.$34 AS InstitutionHasHospital,
  t.$35 AS InstitutionGrantsMedicalDegree,
  t.$36 AS TribalCollege,
  t.$37 AS DegreeOfUrbanization,
  t.$38 AS InstitutionOpenToGeneralPublic,
  t.$39 AS StatusOfInstitution,
  t.$40 AS UnitidForMergedSchools,
  t.$41 AS YearInstitutionWasDeletedFromIPEDS,
  t.$42 AS DateInstitutionClosed,
  t.$43 AS InstitutionIsActive,
  t.$44 AS PrimarilyPostsecondaryIndicator,
  t.$45 AS PostsecondaryInstitutionIndicator,
  t.$46 AS PostsecondaryAndTitleIvInstitutionIndicator,
  t.$47 AS ReportingMethodForStudentCharges,
  t.$48 AS InstitutionalCategory,
  t.$49 AS CarnegieClassification2015Basic,
  t.$50 AS CarnegieClassification2015UndergraduateProgram,
  t.$51 AS CarnegieClassification2015GraduateProgram,
  t.$52 AS CarnegieClassification2015UndergraduateProfile,
  t.$53 AS CarnegieClassification2015EnrollmentProfile,
  t.$54 AS CarnegieClassification2015SizeSetting,
  NULL AS CarnegieClassification2018Basic,
  NULL AS CarnegieClassification2018UndergraduateProgram,
  NULL AS CarnegieClassification2018GraduateProgram,
  NULL AS CarnegieClassification2018UndergraduateProfile,
  NULL AS CarnegieClassification2018EnrollmentProfile,
  NULL AS CarnegieClassification2018SizeSetting,
  t.$55 AS CarnegieClassification20052010Basic,
  t.$56 AS CarnegieClassification2000,
  t.$57 AS LandGrantInstitution,
  t.$58 AS InstitutionSizeCategory,
  t.$59 AS MultiCampusOrganization,
  t.$60 AS NameOfMultiCampusOrganization,
  t.$61 AS IdentificationNumberOfMultiCampusOrganization,
  t.$62 AS CoreBasedStatisticalArea,
  t.$63 AS CBSATypeMetropolitanMicropolitan,
  t.$64 AS CombinedStatisticalArea,
  t.$65 AS NewEnglandCityAndTownArea,
  t.$66 AS FIPSCountyCode,
  t.$67 AS CountyName,
  t.$68 AS StateAnd114thCongressionalDistrictID,
  t.$69 AS LongitudeLocation,
  t.$70 AS LatitudeLocation,
  t.$71 AS NCESGroupCategory,
  t.$72 AS DataFeedbackReport,
  CAST(substring(metadata$filename, 3, 4) AS INTEGER) AcademicYear,
  metadata$filename IngestedFileName,
  metadata$file_row_number RowNumber
```

```
FROM  @IPEDS_HD t
WHERE
  metadata$file_row_number > 1
  AND CAST(substring(metadata$filename, 3, 4) AS INTEGER) < 2018
UNION ALL
SELECT
  t.$1 AS InstitutionIdentifier,
  t.$2 AS InstitutionName,
  t.$3 AS InstitutionNameAlias,
  t.$4 AS StreetAddress,
  t.$5 AS City,
  t.$6 AS State,
  t.$7 AS ZipCode,
  t.$8 AS StateCode,
  t.$9 AS EconomicAnalysisRegions,
  t.$10 AS ChiefAdministrator,
  t.$11 AS ChiefAdministratorTitle,
  t.$12 AS TelephoneNumber,
  t.$13 AS EmployerIdentificationNumber,
  t.$14 AS DunBradstreetNumbers,
  t.$15 AS PostsecondaryEducationIDNumber,
  t.$16 AS TitleIVEligibilityIndicatorCode,
  t.$17 AS InstitutionsWebAddress,
  t.$18 AS AdmissionsOfficeWebAddress,
  t.$19 AS FinancialAidOfficeWebAddress,
  t.$20 AS OnlineApplicationWebAddress,
  t.$21 AS NetPriceCalculatorWebAddress,
  t.$22 AS VeteransMilitaryServiceTuitionPoliciesWebAddress,
  t.$23 AS StudentRightAthleteGraduationRateWebAddress,
  t.$24 AS DisabilityServicesWebAddress,
  t.$25 AS SectorOfInstitution,
  t.$26 AS LevelOfInstitution,
  t.$27 AS ControlOfInstitution,
  t.$28 AS HighestLevelOfOffering,
  t.$29 AS UndergraduateOffering,
  t.$30 AS GraduateOffering,
  t.$31 AS HighestDegreeOffered,
  t.$32 AS DegreeGrantingStatus,
  t.$33 AS HistoricallyBlackCollegeOrUniversity,
  t.$34 AS InstitutionHasHospital,
  t.$35 AS InstitutionGrantsMedicalDegree,
  t.$36 AS TribalCollege,
  t.$37 AS DegreeOfUrbanization,
  t.$38 AS InstitutionOpenToGeneralPublic,
  t.$39 AS StatusOfInstitution,
  t.$40 AS UnitidForMergedSchools,
  t.$41 AS YearInstitutionWasDeletedFromIPEDS,
  t.$42 AS DateInstitutionClosed,
  t.$43 AS InstitutionIsActive,
  t.$44 AS PrimarilyPostsecondaryIndicator,
  t.$45 AS PostsecondaryInstitutionIndicator,
  t.$46 AS PostsecondaryAndTitleIvInstitutionIndicator,
  t.$47 AS ReportingMethodForStudentCharges,
  t.$48 AS InstitutionalCategory,
  t.$55 AS CarnegieClassification2015Basic,
```

```
  NULL AS CarnegieClassification2015UndergraduateProgram,
  NULL AS CarnegieClassification2015GraduateProgram,
  NULL AS CarnegieClassification2015UndergraduateProfile,
  NULL CarnegieClassification2015EnrollmentProfile,
  NULL CarnegieClassification2015SizeSetting,
  t.$49 AS CarnegieClassification2018Basic,
  t.$50 AS CarnegieClassification2018UndergraduateProgram,
  t.$51 AS CarnegieClassification2018GraduateProgram,
  t.$52 AS CarnegieClassification2018UndergraduateProfile,
  t.$53 AS CarnegieClassification2018EnrollmentProfile,
  t.$54 AS CarnegieClassification2018SizeSetting,
  t.$56 AS CarnegieClassification20052010Basic,
  t.$57 AS CarnegieClassification2000,
  t.$58 AS LandGrantInstitution,
  t.$59 AS InstitutionSizeCategory,
  t.$60 AS MultiCampusOrganization,
  t.$61 AS NameOfMultiCampusOrganization,
  t.$62 AS IdentificationNumberOfMultiCampusOrganization,
  t.$63 AS CoreBasedStatisticalArea,
  t.$64 AS CBSATypeMetropolitanMicropolitan,
  t.$65 AS CombinedStatisticalArea,
  t.$66 AS NewEnglandCityAndTownArea,
  t.$67 AS FIPSCountyCode,
  t.$68 AS CountyName,
  t.$69 AS StateAnd114thCongressionalDistrictID,
  t.$70 AS LongitudeLocation,
  t.$71 AS LatitudeLocation,
  t.$72 AS NCESGroupCategory,
  t.$73 AS DataFeedbackReport,
  CAST(substring(metadata$filename, 3, 4) AS INTEGER)
                AcademicYear,
  metadata$filename IngestedFileName,
  metadata$file_row_number RowNumber
FROM
  @IPEDS_HD t
WHERE
  metadata$file_row_number > 1
  AND CAST(substring(metadata$filename, 3, 4) AS INTEGER) > 2017;
```

Modifying stored procedure for the OD table load

Modifying pr_od_AcademicInstitution_Load to accommodate the structural changes in the files 2018 and onwards.

```
CREATE OR REPLACE PROCEDURE pr_od_AcademicInstitution_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_command = `
    INSERT INTO
          Od_AcademicInstitution (
                InstitutionIdentifier,
                InstitutionName,
```

```
                InstitutionNameAlias,
                StreetAddress,
                City,
                State,
                ZipCode,
                StateCode,
                EconomicAnalysisRegions,
                ChiefAdministrator,
                ChiefAdministratorTitle,
                TelephoneNumber,
                EmployerIdentificationNumber,
                DunBradstreetNumbers,
                PostsecondaryEducationIDNumber,
                TitleIVEligibilityIndicatorCode,
                InstitutionsWebAddress,
                AdmissionsOfficeWebAddress,
                FinancialAidOfficeWebAddress,
                OnlineApplicationWebAddress,
                NetPriceCalculatorWebAddress,
                VeteransMilitaryServiceTuitionPoliciesWebAddress,
                StudentRightAthleteGraduationRateWebAddress,
                DisabilityServicesWebAddress,
                SectorOfInstitution,
                LevelOfInstitution,
                ControlOfInstitution,
                HighestLevelOfOffering,
                UndergraduateOffering,
                GraduateOffering,
                HighestDegreeOffered,
                DegreeGrantingStatus,
                HistoricallyBlackCollegeOrUniversity,
                InstitutionHasHospital,
                InstitutionGrantsMedicalDegree,
                TribalCollege,
                DegreeOfUrbanization,
                InstitutionOpenToGeneralPublic,
                StatusOfInstitution,
                UnitidForMergedSchools,
                YearInstitutionWasDeletedFromIPEDS,
                DateInstitutionClosed,
                InstitutionIsActive,
                PrimarilyPostsecondaryIndicator,
                PostsecondaryInstitutionIndicator,
                PostsecondaryAndTitleIvInstitutionIndicator,
                ReportingMethodForStudentCharges,
                InstitutionalCategory,
                CarnegieClassification2015UndergraduateProgram,
                CarnegieClassification2015GraduateProgram,
                CarnegieClassification2015UndergraduateProfile,
                CarnegieClassification2015EnrollmentProfile,
                CarnegieClassification2015SizeSetting,
                CarnegieClassification2018Basic,
                CarnegieClassification2018UndergraduateProgram,
                CarnegieClassification2018GraduateProgram,
                CarnegieClassification2018UndergraduateProfile,
```

```
            CarnegieClassification2018EnrollmentProfile,
            CarnegieClassification2018SizeSetting,
            CarnegieClassification2015Basic,
            CarnegieClassification20052010Basic,
            CarnegieClassification2000,
            LandGrantInstitution,
            InstitutionSizeCategory,
            MultiCampusOrganization,
            NameOfMultiCampusOrganization,
            IdentificationNumberOfMultiCampusOrganization,
            CoreBasedStatisticalArea,
            CBSATypeMetropolitanMicropolitan,
            CombinedStatisticalArea,
            NewEnglandCityAndTownArea,
            FIPSCountyCode,
            CountyName,
            StateAnd114thCongressionalDistrictID,
            LongitudeLocation,
            LatitudeLocation,
            NCESGroupCategory,
            DataFeedbackReport,
            AcademicYear,
            IngestedFileName,
            RowNumber
        )
    SELECT
        InstitutionIdentifier,
        InstitutionName,
        InstitutionNameAlias,
        StreetAddress,
        City,
        State,
        ZipCode,
        StateCode,
        EconomicAnalysisRegions,
        ChiefAdministrator,
        ChiefAdministratorTitle,
        TelephoneNumber,
        EmployerIdentificationNumber,
        DunBradstreetNumbers,
        PostsecondaryEducationIDNumber,
        TitleIVEligibilityIndicatorCode,
        InstitutionsWebAddress,
        AdmissionsOfficeWebAddress,
        FinancialAidOfficeWebAddress,
        OnlineApplicationWebAddress,
        NetPriceCalculatorWebAddress,
        VeteransMilitaryServiceTuitionPoliciesWebAddress,
        StudentRightAthleteGraduationRateWebAddress,
        DisabilityServicesWebAddress,
        SectorOfInstitution,
        LevelOfInstitution,
        ControlOfInstitution,
        HighestLevelOfOffering,
        UndergraduateOffering,
```

```
        GraduateOffering,
        HighestDegreeOffered,
        DegreeGrantingStatus,
        HistoricallyBlackCollegeOrUniversity,
        InstitutionHasHospital,
        InstitutionGrantsMedicalDegree,
        TribalCollege,
        DegreeOfUrbanization,
        InstitutionOpenToGeneralPublic,
        StatusOfInstitution,
        UnitidForMergedSchools,
        YearInstitutionWasDeletedFromIPEDS,
        DateInstitutionClosed,
        InstitutionIsActive,
        PrimarilyPostsecondaryIndicator,
        PostsecondaryInstitutionIndicator,
        PostsecondaryAndTitleIvInstitutionIndicator,
        ReportingMethodForStudentCharges,
        InstitutionalCategory,
        CarnegieClassification2015UndergraduateProgram,
        CarnegieClassification2015GraduateProgram,
        CarnegieClassification2015UndergraduateProfile,
        CarnegieClassification2015EnrollmentProfile,
        CarnegieClassification2015SizeSetting,
        CarnegieClassification2018Basic,
        CarnegieClassification2018UndergraduateProgram,
        CarnegieClassification2018GraduateProgram,
        CarnegieClassification2018UndergraduateProfile,
        CarnegieClassification2018EnrollmentProfile,
        CarnegieClassification2018SizeSetting,
        CarnegieClassification2015Basic,
        CarnegieClassification20052010Basic,
        CarnegieClassification2000,
        LandGrantInstitution,
        InstitutionSizeCategory,
        MultiCampusOrganization,
        NameOfMultiCampusOrganization,
        IdentificationNumberOfMultiCampusOrganization,
        CoreBasedStatisticalArea,
        CBSATypeMetropolitanMicropolitan,
        CombinedStatisticalArea,
        NewEnglandCityAndTownArea,
        FIPSCountyCode,
        CountyName,
        StateAnd114thCongressionalDistrictID,
        LongitudeLocation,
        LatitudeLocation,
        NCESGroupCategory,
        DataFeedbackReport,
        AcademicYear,
        IngestedFileName,
        RowNumber
      FROM
        v_od_AcademicInstitution
  WHERE AcademicYear = ` + YEAR.toString()+`;`
```

```
    try {
        snowflake.execute (
          {sqlText: sql_command}
          );
        return "Succeeded.";  // Return a success/error indicator.
        }
    catch (err) {
        return "Failed: " + err;  // Return a success/error indicator.
        }
$$
;
```

We can load the academic institution data for 2018 and 2019 into the staging table as below.

```
TRUNCATE TABLE od_AcademicInstitution;
CALL pr_od_AcademicInstitution_Load(2018::FLOAT);
CALL pr_od_AcademicInstitution_Load(2019::FLOAT);
```

We can also load multiple files into same OD table and use the specific academic year to continue next process For this book, we would not truncate the OD table from one academic year to the next. In a real life scenario, we would be processing only one year of data at a time.

```
CALL pr_od_AcademicInstitution_Load(2018::FLOAT);
CALL pr_od_AcademicInstitution_Load(2019::FLOAT);
```

Because of the change in the source file, it is necessary to manage metadata as well as changes in ELT and the transformation layer. If data is processed without the change in code, it will result into bad attribute value mapping.

The stored procedure pr_AcademicInstitution_Load is modified to accommodate the new columns added.

```
CREATE OR REPLACE PROCEDURE
                pr_AcademicInstitution_Load(YEAR FLOAT)
  RETURNS STRING
  LANGUAGE javascript
  EXECUTE AS OWNER
  AS
  $$
  var sql_command = `
 MERGE INTO AcademicInstitution T USING (
  SELECT  *  FROM  Od_AcademicInstitution
  WHERE   ACADEMICYEAR = ` + YEAR.toString() +`
) S ON T.InstitutionIdentifier
                = S.InstitutionIdentifier
WHEN MATCHED
AND (
  IFNULL(T.InstitutionName, '')
                <> IFNULL(S.InstitutionName, '')
  OR IFNULL(T.InstitutionNameAlias, '')
                <> IFNULL(S.InstitutionNameAlias, '')
  OR IFNULL(T.StreetAddress, '')
                <> IFNULL(S.StreetAddress, '')
  OR IFNULL(T.City, '') <> IFNULL(S.City, '')
```

```
  OR IFNULL(T.State, '') <> IFNULL(S.State, '')
  OR IFNULL(T.ZipCode, '') <> IFNULL(S.ZipCode, '')
  OR IFNULL(T.StateCode, 0) <> IFNULL(S.StateCode, 0)
  OR IFNULL(T.EconomicAnalysisRegions, 0)
                <> IFNULL(S.EconomicAnalysisRegions, 0)
  OR IFNULL(T.ChiefAdministrator, '')
                <> IFNULL(S.ChiefAdministrator, '')
  OR IFNULL(T.ChiefAdministratorTitle, '')
                <> IFNULL(S.ChiefAdministratorTitle, '')
  OR IFNULL(T.TelephoneNumber, '')
                <> IFNULL(S.TelephoneNumber, '')
  OR IFNULL(T.EmployerIdentificationNumber, '')
                <> IFNULL(S.EmployerIdentificationNumber, '')
  OR IFNULL(T.DunBradstreetNumbers, '')
                <> IFNULL(S.DunBradstreetNumbers, '')
  OR IFNULL(T.PostsecondaryEducationIDNumber, '')
                <> IFNULL(S.PostsecondaryEducationIDNumber, '')
  OR IFNULL(T.TitleIVEligibilityIndicatorCode, 0)
                <> IFNULL(S.TitleIVEligibilityIndicatorCode, 0)
  OR IFNULL(T.SectorOfInstitution, 0)
                <> IFNULL(S.SectorOfInstitution, 0)
  OR IFNULL(T.LevelOfInstitution, 0)
                <> IFNULL(S.LevelOfInstitution, 0)
  OR IFNULL(T.ControlOfInstitution, 0)
                <> IFNULL(S.ControlOfInstitution, 0)
  OR IFNULL(T.HighestLevelOfOffering, 0)
                <> IFNULL(S.HighestLevelOfOffering, 0)
  OR IFNULL(T.UndergraduateOffering, 0)
                <> IFNULL(S.UndergraduateOffering, 0)
  OR IFNULL(T.GraduateOffering, 0)
                <> IFNULL(S.GraduateOffering, 0)
  OR IFNULL(T.HighestDegreeOffered, 0)
                <> IFNULL(S.HighestDegreeOffered, 0)
  OR IFNULL(T.DegreeGrantingStatus, 0)
                <> IFNULL(S.DegreeGrantingStatus, 0)
  OR IFNULL(T.HistoricallyBlackCollegeOrUniversity, 0)
                <> IFNULL(S.HistoricallyBlackCollegeOrUniversity, 0)
  OR IFNULL(T.InstitutionHasHospital, 0)
                <> IFNULL(S.InstitutionHasHospital, 0)
  OR IFNULL(T.InstitutionGrantsMedicalDegree, 0)
                <> IFNULL(S.InstitutionGrantsMedicalDegree, 0)
  OR IFNULL(T.TribalCollege, 0)
                <> IFNULL(S.TribalCollege, 0)
  OR IFNULL(T.DegreeOfUrbanization, 0)
                <> IFNULL(S.DegreeOfUrbanization, 0)
  OR IFNULL(T.InstitutionOpenToGeneralPublic, 0)
                <> IFNULL(S.InstitutionOpenToGeneralPublic, 0)
  OR IFNULL(T.StatusOfInstitution, '')
                <> IFNULL(S.StatusOfInstitution, '')
  OR IFNULL(T.UnitidForMergedSchools, 0)
                <> IFNULL(S.UnitidForMergedSchools, 0)
  OR IFNULL(T.YearInstitutionWasDeletedFromIPEDS, 0)
                <> IFNULL(S.YearInstitutionWasDeletedFromIPEDS, 0)
  OR IFNULL(T.DateInstitutionClosed, '')
                <> IFNULL(S.DateInstitutionClosed, '')
```

```
  OR IFNULL(T.InstitutionIsActive, 0)
                <> IFNULL(S.InstitutionIsActive, 0)
  OR IFNULL(T.PrimarilyPostsecondaryIndicator, 0)
                <> IFNULL(S.PrimarilyPostsecondaryIndicator, 0)
  OR IFNULL(T.PostsecondaryInstitutionIndicator, 0)
                <> IFNULL(S.PostsecondaryInstitutionIndicator, 0)
  OR IFNULL(T.PostsecondaryAndTitleIvInstitutionIndicator, 0)
        <> IFNULL(S.PostsecondaryAndTitleIvInstitutionIndicator, 0)
  OR IFNULL(T.ReportingMethodForStudentCharges, 0)
                <> IFNULL(S.ReportingMethodForStudentCharges, 0)
  OR IFNULL(T.InstitutionalCategory, 0)
                <> IFNULL(S.InstitutionalCategory, 0)
  OR IFNULL(T.LandGrantInstitution, 0)
                <> IFNULL(S.LandGrantInstitution, 0)
  OR IFNULL(T.InstitutionSizeCategory, 0)
                <> IFNULL(S.InstitutionSizeCategory, 0)
  OR IFNULL(T.MultiCampusOrganization, 0)
                <> IFNULL(S.MultiCampusOrganization, 0)
  OR IFNULL(T.NameOfMultiCampusOrganization, '')
                <> IFNULL(S.NameOfMultiCampusOrganization, '')
  OR IFNULL(T.IdentificationNumberOfMultiCampusOrganization, ''  )
        <> IFNULL(S.IdentificationNumberOfMultiCampusOrganization,'')
  OR IFNULL(T.CoreBasedStatisticalArea, 0)
                <> IFNULL(S.CoreBasedStatisticalArea, 0)
  OR IFNULL(T.CBSATypeMetropolitanMicropolitan, 0)
                <> IFNULL(S.CBSATypeMetropolitanMicropolitan, 0)
  OR IFNULL(T.CombinedStatisticalArea, 0)
                <> IFNULL(S.CombinedStatisticalArea, 0)
  OR IFNULL(T.NewEnglandCityAndTownArea, 0)
                <> IFNULL(S.NewEnglandCityAndTownArea, 0)
  OR IFNULL(T.FIPSCountyCode, 0) <> IFNULL(S.FIPSCountyCode, 0)
  OR IFNULL(T.CountyName, '') <> IFNULL(S.CountyName, '')
  OR IFNULL(T.StateAnd114thCongressionalDistrictID, 0)
                <> IFNULL(S.StateAnd114thCongressionalDistrictID, 0)
  OR IFNULL(T.LongitudeLocation, 0.0)
                <> IFNULL(S.LongitudeLocation, 0.0)
  OR IFNULL(T.LatitudeLocation, 0.0)
                <> IFNULL(S.LatitudeLocation, 0.0)
  OR IFNULL(T.NCESGroupCategory, 0)
                <> IFNULL(S.NCESGroupCategory, 0)
  OR IFNULL(T.DataFeedbackReport, 0)
                <> IFNULL(S.DataFeedbackReport, 0)
  OR CarnegieClassification  <> OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'CarnegieClassification2000' ,
                S.CarnegieClassification2000,
        'CarnegieClassification20052010Basic' ,
                S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic' ,
                S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram' ,
                S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram' ,
                S.CarnegieClassification2015GraduateProgram,
```

```
        'CarnegieClassification2015UndergraduateProfile' ,
                S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile' ,
                S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting' ,
                S.CarnegieClassification2015SizeSetting ,
        'CarnegieClassification2018Basic',
                S.CarnegieClassification2018Basic,
        'CarnegieClassification2018UndergraduateProgram' ,
                S.CarnegieClassification2018UndergraduateProgram,
        'CarnegieClassification2018GraduateProgram',
                S.CarnegieClassification2018GraduateProgram,
        'CarnegieClassification2018UndergraduateProfile' ,
                S.CarnegieClassification2018UndergraduateProfile,
        'CarnegieClassification2018EnrollmentProfile',
                S.CarnegieClassification2018EnrollmentProfile,
        'CarnegieClassification2018SizeSetting' ,
                S.CarnegieClassification2018SizeSetting
                        )
  OR WebAddress<>OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'InstitutionsWebAddress',
                S.InstitutionsWebAddress,
        'AdmissionsOfficeWebAddress',
                S.AdmissionsOfficeWebAddress,
        'FinancialAidOfficeWebAddress',
                S.FinancialAidOfficeWebAddress,
        'OnlineApplicationWebAddress',
                S.OnlineApplicationWebAddress,
        'NetPriceCalculatorWebAddress',
                S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress',
                S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress',
                S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress',
                S.DisabilityServicesWebAddress
                        )

) THEN
UPDATE
SET
  InstitutionIdentifier = S.InstitutionIdentifier,
  InstitutionName = S.InstitutionName,
  InstitutionNameAlias = S.InstitutionNameAlias,
  StreetAddress = S.StreetAddress,
  City = S.City,
  State = S.State,
  ZipCode = S.ZipCode,
  StateCode = S.StateCode,
  EconomicAnalysisRegions = S.EconomicAnalysisRegions,
  ChiefAdministrator = S.ChiefAdministrator,
  ChiefAdministratorTitle = S.ChiefAdministratorTitle,
  TelephoneNumber = S.TelephoneNumber,
```

```
EmployerIdentificationNumber
              = S.EmployerIdentificationNumber,
DunBradstreetNumbers = S.DunBradstreetNumbers,
PostsecondaryEducationIDNumber
              = S.PostsecondaryEducationIDNumber,
TitleIVEligibilityIndicatorCode
              = S.TitleIVEligibilityIndicatorCode,
SectorOfInstitution = S.SectorOfInstitution,
LevelOfInstitution = S.LevelOfInstitution,
ControlOfInstitution = S.ControlOfInstitution,
HighestLevelOfOffering = S.HighestLevelOfOffering,
UndergraduateOffering = S.UndergraduateOffering,
GraduateOffering = S.GraduateOffering,
HighestDegreeOffered = S.HighestDegreeOffered,
DegreeGrantingStatus = S.DegreeGrantingStatus,
HistoricallyBlackCollegeOrUniversity
              = S.HistoricallyBlackCollegeOrUniversity,
InstitutionHasHospital = S.InstitutionHasHospital,
InstitutionGrantsMedicalDegree
              = S.InstitutionGrantsMedicalDegree,
TribalCollege = S.TribalCollege,
DegreeOfUrbanization = S.DegreeOfUrbanization,
InstitutionOpenToGeneralPublic
              = S.InstitutionOpenToGeneralPublic,
StatusOfInstitution = S.StatusOfInstitution,
UnitidForMergedSchools = S.UnitidForMergedSchools,
YearInstitutionWasDeletedFromIPEDS
              = S.YearInstitutionWasDeletedFromIPEDS,
DateInstitutionClosed = S.DateInstitutionClosed,
InstitutionIsActive = S.InstitutionIsActive,
PrimarilyPostsecondaryIndicator
              = S.PrimarilyPostsecondaryIndicator,
PostsecondaryInstitutionIndicator
              = S.PostsecondaryInstitutionIndicator,
PostsecondaryAndTitleIvInstitutionIndicator
              = S.PostsecondaryAndTitleIvInstitutionIndicator,
ReportingMethodForStudentCharges
              = S.ReportingMethodForStudentCharges,
InstitutionalCategory = S.InstitutionalCategory,
LandGrantInstitution = S.LandGrantInstitution,
InstitutionSizeCategory = S.InstitutionSizeCategory,
MultiCampusOrganization = S.MultiCampusOrganization,
NameOfMultiCampusOrganization
              = S.NameOfMultiCampusOrganization,
IdentificationNumberOfMultiCampusOrganization
              = S.IdentificationNumberOfMultiCampusOrganization,
CoreBasedStatisticalArea
              = S.CoreBasedStatisticalArea,
CBSATypeMetropolitanMicropolitan
              = S.CBSATypeMetropolitanMicropolitan,
CombinedStatisticalArea = S.CombinedStatisticalArea,
NewEnglandCityAndTownArea = S.NewEnglandCityAndTownArea,
FIPSCountyCode = S.FIPSCountyCode,
CountyName = S.CountyName,
StateAnd114thCongressionalDistrictID
```

```
                = S.StateAnd114thCongressionalDistrictID,
  LongitudeLocation = S.LongitudeLocation,
  LatitudeLocation = S.LatitudeLocation,
  NCESGroupCategory = S.NCESGroupCategory,
  DataFeedbackReport = S.DataFeedbackReport,
  AcademicYear=S.AcademicYear,
  CarnegieClassification =OBJECT_CONSTRUCT (
  'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'CarnegieClassification2000' ,
                S.CarnegieClassification2000,
        'CarnegieClassification20052010Basic' ,
                S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic' ,
                S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram' ,
                S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram' ,
                S.CarnegieClassification2015GraduateProgram,
        'CarnegieClassification2015UndergraduateProfile' ,
                S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile' ,
                S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting' ,
                S.CarnegieClassification2015SizeSetting ,
        'CarnegieClassification2018Basic',
                S.CarnegieClassification2018Basic,
        'CarnegieClassification2018UndergraduateProgram' ,
                S.CarnegieClassification2018UndergraduateProgram,
        'CarnegieClassification2018GraduateProgram',
                S.CarnegieClassification2018GraduateProgram,
        'CarnegieClassification2018UndergraduateProfile' ,
                S.CarnegieClassification2018UndergraduateProfile,
        'CarnegieClassification2018EnrollmentProfile',
                S.CarnegieClassification2018EnrollmentProfile,
        'CarnegieClassification2018SizeSetting' ,
                S.CarnegieClassification2018SizeSetting
                        ),
       WebAddress=OBJECT_CONSTRUCT (
       'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
       'InstitutionsWebAddress',
                S.InstitutionsWebAddress,
       'AdmissionsOfficeWebAddress',
                S.AdmissionsOfficeWebAddress,
       'FinancialAidOfficeWebAddress',
                S.FinancialAidOfficeWebAddress,
       'OnlineApplicationWebAddress',
                S.OnlineApplicationWebAddress,
       'NetPriceCalculatorWebAddress',
                S.NetPriceCalculatorWebAddress,
       'VeteransMilitaryServiceTuitionPoliciesWebAddress',
                S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
       'StudentRightAthleteGraduationRateWebAddress',
                S.StudentRightAthleteGraduationRateWebAddress,
```

```
        'DisabilityServicesWebAddress',
               S.DisabilityServicesWebAddress
                     ),
  RecordUpdateDateTime = CURRENT_TIMESTAMP
WHEN NOT MATCHED THEN
       INSERT   (
    InstitutionIdentifier,
    InstitutionName,
    InstitutionNameAlias,
    StreetAddress,
    City,
    State,
    ZipCode,
    StateCode,
    EconomicAnalysisRegions,
    ChiefAdministrator,
    ChiefAdministratorTitle,
    TelephoneNumber,
    EmployerIdentificationNumber,
    DunBradstreetNumbers,
    PostsecondaryEducationIDNumber,
    TitleIVEligibilityIndicatorCode,
    SectorOfInstitution,
    LevelOfInstitution,
    ControlOfInstitution,
    HighestLevelOfOffering,
    UndergraduateOffering,
    GraduateOffering,
    HighestDegreeOffered,
    DegreeGrantingStatus,
    HistoricallyBlackCollegeOrUniversity,
    InstitutionHasHospital,
    InstitutionGrantsMedicalDegree,
    TribalCollege,
    DegreeOfUrbanization,
    InstitutionOpenToGeneralPublic,
    StatusOfInstitution,
    UnitidForMergedSchools,
    YearInstitutionWasDeletedFromIPEDS,
    DateInstitutionClosed,
    InstitutionIsActive,
    PrimarilyPostsecondaryIndicator,
    PostsecondaryInstitutionIndicator,
    PostsecondaryAndTitleIvInstitutionIndicator,
    ReportingMethodForStudentCharges,
    InstitutionalCategory,
    LandGrantInstitution,
    InstitutionSizeCategory,
    MultiCampusOrganization,
    NameOfMultiCampusOrganization,
    IdentificationNumberOfMultiCampusOrganization,
    CoreBasedStatisticalArea,
    CBSATypeMetropolitanMicropolitan,
    CombinedStatisticalArea,
    NewEnglandCityAndTownArea,
```

```
    FIPSCountyCode,
    CountyName,
    StateAnd114thCongressionalDistrictID,
    LongitudeLocation,
    LatitudeLocation,
    NCESGroupCategory,
    DataFeedbackReport,
    AcademicYear,
        CarnegieClassification,
        WebAddress,
    RecordCreateDateTime)
VALUES  (
    S.InstitutionIdentifier,
    S.InstitutionName,
    S.InstitutionNameAlias,
    S.StreetAddress,
    S.City,
    S.State,
    S.ZipCode,
    S.StateCode,
    S.EconomicAnalysisRegions,
    S.ChiefAdministrator,
    S.ChiefAdministratorTitle,
    S.TelephoneNumber,
    S.EmployerIdentificationNumber,
    S.DunBradstreetNumbers,
    S.PostsecondaryEducationIDNumber,
    S.TitleIVEligibilityIndicatorCode,
    S.SectorOfInstitution,
    S.LevelOfInstitution,
    S.ControlOfInstitution,
    S.HighestLevelOfOffering,
    S.UndergraduateOffering,
    S.GraduateOffering,
    S.HighestDegreeOffered,
    S.DegreeGrantingStatus,
    S.HistoricallyBlackCollegeOrUniversity,
    S.InstitutionHasHospital,
    S.InstitutionGrantsMedicalDegree,
    S.TribalCollege,
    S.DegreeOfUrbanization,
    S.InstitutionOpenToGeneralPublic,
    S.StatusOfInstitution,
    S.UnitidForMergedSchools,
    S.YearInstitutionWasDeletedFromIPEDS,
    S.DateInstitutionClosed,
    S.InstitutionIsActive,
    S.PrimarilyPostsecondaryIndicator,
    S.PostsecondaryInstitutionIndicator,
    S.PostsecondaryAndTitleIvInstitutionIndicator,
    S.ReportingMethodForStudentCharges,
    S.InstitutionalCategory,
    S.LandGrantInstitution,
    S.InstitutionSizeCategory,
    S.MultiCampusOrganization,
```

```
    S.NameOfMultiCampusOrganization,
    S.IdentificationNumberOfMultiCampusOrganization,
    S.CoreBasedStatisticalArea,
    S.CBSATypeMetropolitanMicropolitan,
    S.CombinedStatisticalArea,
    S.NewEnglandCityAndTownArea,
    S.FIPSCountyCode,
    S.CountyName,
    S.StateAnd114thCongressionalDistrictID,
    S.LongitudeLocation,
    S.LatitudeLocation,
    S.NCESGroupCategory,
    S.DataFeedbackReport,
    S.AcademicYear,
        OBJECT_CONSTRUCT (
        'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'CarnegieClassification2000' ,
                S.CarnegieClassification2000,
        'CarnegieClassification20052010Basic' ,
                S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic' ,
                S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram' ,
                S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram' ,
                S.CarnegieClassification2015GraduateProgram,
        'CarnegieClassification2015UndergraduateProfile' ,
                S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile' ,
                S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting' ,
                S.CarnegieClassification2015SizeSetting ,
        'CarnegieClassification2018Basic',
                S.CarnegieClassification2018Basic,
        'CarnegieClassification2018UndergraduateProgram' ,
                S.CarnegieClassification2018UndergraduateProgram,
        'CarnegieClassification2018GraduateProgram',
                S.CarnegieClassification2018GraduateProgram,
        'CarnegieClassification2018UndergraduateProfile' ,
                S.CarnegieClassification2018UndergraduateProfile,
        'CarnegieClassification2018EnrollmentProfile',
                S.CarnegieClassification2018EnrollmentProfile,
        'CarnegieClassification2018SizeSetting' ,
                S.CarnegieClassification2018SizeSetting
                        ) ,
        OBJECT_CONSTRUCT (

        'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'InstitutionsWebAddress',
                S.InstitutionsWebAddress,
        'AdmissionsOfficeWebAddress',
                S.AdmissionsOfficeWebAddress,
        'FinancialAidOfficeWebAddress',
```

```
                S.FinancialAidOfficeWebAddress,
        'OnlineApplicationWebAddress',
                S.OnlineApplicationWebAddress,
        'NetPriceCalculatorWebAddress',
                S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress',
                S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress',
                S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress',
                S.DisabilityServicesWebAddress
                        ),
    CURRENT_TIMESTAMP
        );
`
try {
    snowflake.execute (
      {sqlText: sql_command}
      );
    return "Succeeded.";
    }
  catch (err) {
    return "Failed: " + err;
    }
  $$
  ;
```

We are able to ingest all the files from 2017 through 2019 with the above schema drift code changes

```
    TRUNCATE TABLE od_AcademicInstitution;
    CALL pr_od_AcademicInstitution_Load(2017::FLOAT);
    CALL pr_od_AcademicInstitution_Load(2018::FLOAT);
    CALL pr_od_AcademicInstitution_Load(2019::FLOAT);
```

We can process dimension by calling the stored procedure as below.

```
    CALL pr_AcademicInstitution_Load(2017::FLOAT);
    CALL pr_AcademicInstitution_Load(2018::FLOAT);
    CALL pr_AcademicInstitution_Load(2019::FLOAT);
```

8.9 MERGE using hash key

In the above stored procedure, we joined the source and target table using the business key (InstitutionIdentifier). When a matching record is found, then we compare individual columns between source and target tables to find any column values that are changed in the source.

In this section, we are trying to compare source and target column values by using hash keys. Hash keys are created using hash functions. In a hash function, a plain text of arbitrary size is mapped to fixed-size values called hash keys or bytes. Hash functions use various hashing algorithms for generating hash keys. MD5, SHA1, SHA2 etc. are some of the commonly used algorithms.

We created a custom function called **Hash_Key** which accepts an array of text values. These values are concatenated using another function **Concat_Columns** and returns a string value. This concatenated string is passed to the hash function SHA1 to generate the hash key out of it. Function Concat_Columns is created as below.

```
CREATE OR REPLACE FUNCTION
    ConcatColumns(COLUMNLIST array)
    RETURNS varchar
    LANGUAGE javascript
  AS
  $$
    function return_column_string(ColumnArray) {
      var ColumnString = '';
      var arrayLength = ColumnArray.length;
      for (var i = 0; i < arrayLength; i++) {
        if (i>0) ColumnString += ';'
        ColumnString += typeof(ColumnArray[i])
                        == 'undefined' ? '' : ColumnArray[i] ;
      }
      return ColumnString.toUpperCase().trim();
    }
  return return_column_string(COLUMNLIST);
  $$
;
```

Here are the scripts for creating the function Hash_Key.

```
CREATE OR REPLACE FUNCTION
    Hash_Key(SCDKeys array)
    RETURNS VARCHAR(50)
  as
  $$
    UPPER(SHA1(ConcatColumns(SCDKeys)))
  $$
;
```

We created a hash key by combining all column values of each row in the source and target tables. And, then these hash keys are compared to check if any change occurred in the source data. If there is a change, the corresponding target record is updated with the source record.

The modified stored procedure is given below.

```
CREATE OR REPLACE PROCEDURE
                pr_AcademicInstitutionMergeHashByte_Load(YEAR FLOAT)
  RETURNS STRING
  LANGUAGE javascript
  EXECUTE AS OWNER
  AS
  $$
  var sql_command = `
        MERGE INTO AcademicInstitution T USING (
        SELECT  *  FROM  Od_AcademicInstitution
        WHERE   ACADEMICYEAR = ` + YEAR.toString() +`
        ) S ON T.InstitutionIdentifier = S.InstitutionIdentifier
        WHEN MATCHED
                AND ( HashKey(ARRAY_CONSTRUCT(
        T.InstitutionName,
        T.InstitutionNameAlias,
        T.StreetAddress,
```

```
        T.City,
        T.State,
        T.ZipCode,
        T.StateCode,
        T.EconomicAnalysisRegions,
        T.ChiefAdministrator,
        T.ChiefAdministratorTitle,
        T.TelephoneNumber,
        T.EmployerIdentificationNumber,
        T.DunBradstreetNumbers,
        T.PostsecondaryEducationIDNumber,
        T.TitleIVEligibilityIndicatorCode,
        T.SectorOfInstitution,
        T.LevelOfInstitution,
        T.ControlOfInstitution,
        T.HighestLevelOfOffering,
        T.UndergraduateOffering,
        T.GraduateOffering,
        T.HighestDegreeOffered,
        T.DegreeGrantingStatus,
        T.HistoricallyBlackCollegeOrUniversity,
        T.InstitutionHasHospital,
        T.InstitutionGrantsMedicalDegree,
        T.TribalCollege,
        T.DegreeOfUrbanization,
        T.InstitutionOpenToGeneralPublic,
        T.StatusOfInstitution,
        T.UnitidForMergedSchools,
        T.YearInstitutionWasDeletedFromIPEDS,
        T.DateInstitutionClosed,
        T.InstitutionIsActive,
        T.PrimarilyPostsecondaryIndicator,
        T.PostsecondaryInstitutionIndicator,
        T.PostsecondaryAndTitleIvInstitutionIndicator,
        T.ReportingMethodForStudentCharges,
        T.InstitutionalCategory,
        T.LandGrantInstitution,
        T.InstitutionSizeCategory,
        T.MultiCampusOrganization,
        T.NameOfMultiCampusOrganization,
        T.IdentificationNumberOfMultiCampusOrganization,
        T.CoreBasedStatisticalArea,
        T.CBSATypeMetropolitanMicropolitan,
        T.CombinedStatisticalArea,
        T.NewEnglandCityAndTownArea,
        T.FIPSCountyCode,
        T.CountyName,
        T.StateAnd114thCongressionalDistrictID,
        T.LongitudeLocation,
        T.LatitudeLocation,
        T.NCESGroupCategory,
        T.DataFeedbackReport,
        T.CarnegieClassification,
        T.WebAddress)) <> HashKey(ARRAY_CONSTRUCT(
               S.InstitutionName,
```

```
            S.InstitutionNameAlias,
            S.StreetAddress,
            S.City,
            S.State,
            S.ZipCode,
            S.StateCode,
            S.EconomicAnalysisRegions,
            S.ChiefAdministrator,
            S.ChiefAdministratorTitle,
            S.TelephoneNumber,
            S.EmployerIdentificationNumber,
            S.DunBradstreetNumbers,
            S.PostsecondaryEducationIDNumber,
            S.TitleIVEligibilityIndicatorCode,
            S.SectorOfInstitution,
            S.LevelOfInstitution,
            S.ControlOfInstitution,
            S.HighestLevelOfOffering,
            S.UndergraduateOffering,
            S.GraduateOffering,
            S.HighestDegreeOffered,
            S.DegreeGrantingStatus,
            S.HistoricallyBlackCollegeOrUniversity,
            S.InstitutionHasHospital,
            S.InstitutionGrantsMedicalDegree,
            S.TribalCollege,
            S.DegreeOfUrbanization,
            S.InstitutionOpenToGeneralPublic,
            S.StatusOfInstitution,
            S.UnitidForMergedSchools,
            S.YearInstitutionWasDeletedFromIPEDS,
            S.DateInstitutionClosed,
            S.InstitutionIsActive,
            S.PrimarilyPostsecondaryIndicator,
            S.PostsecondaryInstitutionIndicator,
            S.PostsecondaryAndTitleIvInstitutionIndicator,
            S.ReportingMethodForStudentCharges,
            S.InstitutionalCategory,
            S.LandGrantInstitution,
            S.InstitutionSizeCategory,
            S.MultiCampusOrganization,
            S.NameOfMultiCampusOrganization,
            S.IdentificationNumberOfMultiCampusOrganization,
            S.CoreBasedStatisticalArea,
            S.CBSATypeMetropolitanMicropolitan,
            S.CombinedStatisticalArea,
            S.NewEnglandCityAndTownArea,
            S.FIPSCountyCode,
            S.CountyName,
            S.StateAnd114thCongressionalDistrictID,
            S.LongitudeLocation,
            S.LatitudeLocation,
            S.NCESGroupCategory,
            S.DataFeedbackReport,
        OBJECT_CONSTRUCT (
```

```
        'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'CarnegieClassification2000' ,
                S.CarnegieClassification2000,
        'CarnegieClassification20052010Basic' ,
                S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic' ,
                S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram' ,
                S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram' ,
                S.CarnegieClassification2015GraduateProgram,
        'CarnegieClassification2015UndergraduateProfile' ,
                S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile' ,
                S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting' ,
                S.CarnegieClassification2015SizeSetting ,
        'CarnegieClassification2018Basic',
                S.CarnegieClassification2018Basic,
        'CarnegieClassification2018UndergraduateProgram' ,
                S.CarnegieClassification2018UndergraduateProgram,
        'CarnegieClassification2018GraduateProgram',
                S.CarnegieClassification2018GraduateProgram,
        'CarnegieClassification2018UndergraduateProfile' ,
                S.CarnegieClassification2018UndergraduateProfile,
        'CarnegieClassification2018EnrollmentProfile',
                S.CarnegieClassification2018EnrollmentProfile,
        'CarnegieClassification2018SizeSetting' ,
                S.CarnegieClassification2018SizeSetting
                                ) ,
        OBJECT_CONSTRUCT (

        'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'InstitutionsWebAddress',
                S.InstitutionsWebAddress,
        'AdmissionsOfficeWebAddress',
                S.AdmissionsOfficeWebAddress,
        'FinancialAidOfficeWebAddress',
                S.FinancialAidOfficeWebAddress,
        'OnlineApplicationWebAddress',
                S.OnlineApplicationWebAddress,
        'NetPriceCalculatorWebAddress',
                S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress',
                S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress',
                S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress',
                S.DisabilityServicesWebAddress

                                )
        ))) THEN
UPDATE
```

```
SET
  InstitutionIdentifier = S.InstitutionIdentifier,
  InstitutionName = S.InstitutionName,
  InstitutionNameAlias = S.InstitutionNameAlias,
  StreetAddress = S.StreetAddress,
  City = S.City,
  State = S.State,
  ZipCode = S.ZipCode,
  StateCode = S.StateCode,
  EconomicAnalysisRegions = S.EconomicAnalysisRegions,
  ChiefAdministrator = S.ChiefAdministrator,
  ChiefAdministratorTitle = S.ChiefAdministratorTitle,
  TelephoneNumber = S.TelephoneNumber,
  EmployerIdentificationNumber = S.EmployerIdentificationNumber,
  DunBradstreetNumbers = S.DunBradstreetNumbers,
  PostsecondaryEducationIDNumber = S.PostsecondaryEducationIDNumber,
  TitleIVEligibilityIndicatorCode = S.TitleIVEligibilityIndicatorCode,
  SectorOfInstitution = S.SectorOfInstitution,
  LevelOfInstitution = S.LevelOfInstitution,
  ControlOfInstitution = S.ControlOfInstitution,
  HighestLevelOfOffering = S.HighestLevelOfOffering,
  UndergraduateOffering = S.UndergraduateOffering,
  GraduateOffering = S.GraduateOffering,
  HighestDegreeOffered = S.HighestDegreeOffered,
  DegreeGrantingStatus = S.DegreeGrantingStatus,
  HistoricallyBlackCollegeOrUniversity
        = S.HistoricallyBlackCollegeOrUniversity,
  InstitutionHasHospital = S.InstitutionHasHospital,
  InstitutionGrantsMedicalDegree = S.InstitutionGrantsMedicalDegree,
  TribalCollege = S.TribalCollege,
  DegreeOfUrbanization = S.DegreeOfUrbanization,
  InstitutionOpenToGeneralPublic
        = S.InstitutionOpenToGeneralPublic,
  StatusOfInstitution = S.StatusOfInstitution,
  UnitidForMergedSchools = S.UnitidForMergedSchools,
  YearInstitutionWasDeletedFromIPEDS
        = S.YearInstitutionWasDeletedFromIPEDS,
  DateInstitutionClosed = S.DateInstitutionClosed,
  InstitutionIsActive = S.InstitutionIsActive,
  PrimarilyPostsecondaryIndicator = S.PrimarilyPostsecondaryIndicator,
  PostsecondaryInstitutionIndicator = S.PostsecondaryInstitutionIndicator,
  PostsecondaryAndTitleIvInstitutionIndicator
        = S.PostsecondaryAndTitleIvInstitutionIndicator,
  ReportingMethodForStudentCharges = S.ReportingMethodForStudentCharges,
  InstitutionalCategory = S.InstitutionalCategory,
  LandGrantInstitution = S.LandGrantInstitution,
  InstitutionSizeCategory = S.InstitutionSizeCategory,
  MultiCampusOrganization = S.MultiCampusOrganization,
  NameOfMultiCampusOrganization = S.NameOfMultiCampusOrganization,
  IdentificationNumberOfMultiCampusOrganization
        = S.IdentificationNumberOfMultiCampusOrganization,
  CoreBasedStatisticalArea = S.CoreBasedStatisticalArea,
  CBSATypeMetropolitanMicropolitan
        = S.CBSATypeMetropolitanMicropolitan,
  CombinedStatisticalArea = S.CombinedStatisticalArea,
```

```
NewEnglandCityAndTownArea = S.NewEnglandCityAndTownArea,
FIPSCountyCode = S.FIPSCountyCode,
CountyName = S.CountyName,
StateAnd114thCongressionalDistrictID
      = S.StateAnd114thCongressionalDistrictID,
LongitudeLocation = S.LongitudeLocation,
LatitudeLocation = S.LatitudeLocation,
NCESGroupCategory = S.NCESGroupCategory,
DataFeedbackReport = S.DataFeedbackReport,
AcademicYear=S.AcademicYear,
CarnegieClassification =OBJECT_CONSTRUCT (
'AcademicInstitutionIdentifier',
              S.InstitutionIdentifier,
      'CarnegieClassification2000' ,
              S.CarnegieClassification2000,
      'CarnegieClassification20052010Basic' ,
              S.CarnegieClassification20052010Basic,
      'CarnegieClassification2015Basic' ,
              S.CarnegieClassification2015Basic,
      'CarnegieClassification2015UndergraduateProgram' ,
              S.CarnegieClassification2015UndergraduateProgram,
      'CarnegieClassification2015GraduateProgram' ,
              S.CarnegieClassification2015GraduateProgram,
      'CarnegieClassification2015UndergraduateProfile' ,
              S.CarnegieClassification2015UndergraduateProfile,
      'CarnegieClassification2015EnrollmentProfile' ,
              S.CarnegieClassification2015EnrollmentProfile,
      'CarnegieClassification2015SizeSetting' ,
              S.CarnegieClassification2015SizeSetting ,
      'CarnegieClassification2018Basic',
              S.CarnegieClassification2018Basic,
      'CarnegieClassification2018UndergraduateProgram' ,
              S.CarnegieClassification2018UndergraduateProgram,
      'CarnegieClassification2018GraduateProgram',
              S.CarnegieClassification2018GraduateProgram,
      'CarnegieClassification2018UndergraduateProfile' ,
              S.CarnegieClassification2018UndergraduateProfile,
      'CarnegieClassification2018EnrollmentProfile',
              S.CarnegieClassification2018EnrollmentProfile,
      'CarnegieClassification2018SizeSetting' ,
              S.CarnegieClassification2018SizeSetting

 ),
      WebAddress=OBJECT_CONSTRUCT (

      'AcademicInstitutionIdentifier',
              S.InstitutionIdentifier,
      'InstitutionsWebAddress',
              S.InstitutionsWebAddress,
      'AdmissionsOfficeWebAddress',
              S.AdmissionsOfficeWebAddress,
      'FinancialAidOfficeWebAddress',
              S.FinancialAidOfficeWebAddress,
      'OnlineApplicationWebAddress',
              S.OnlineApplicationWebAddress,
```

```
        'NetPriceCalculatorWebAddress',
                S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress',
                S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress',
                S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress',
                S.DisabilityServicesWebAddress

   ),
  RecordUpdateDateTime = CURRENT_TIMESTAMP
WHEN NOT MATCHED THEN
        INSERT   (
        InstitutionIdentifier,
        InstitutionName,
        InstitutionNameAlias,
        StreetAddress,
        City,
        State,
        ZipCode,
        StateCode,
        EconomicAnalysisRegions,
        ChiefAdministrator,
        ChiefAdministratorTitle,
        TelephoneNumber,
        EmployerIdentificationNumber,
        DunBradstreetNumbers,
        PostsecondaryEducationIDNumber,
        TitleIVEligibilityIndicatorCode,
        SectorOfInstitution,
        LevelOfInstitution,
        ControlOfInstitution,
        HighestLevelOfOffering,
        UndergraduateOffering,
        GraduateOffering,
        HighestDegreeOffered,
        DegreeGrantingStatus,
        HistoricallyBlackCollegeOrUniversity,
        InstitutionHasHospital,
        InstitutionGrantsMedicalDegree,
        TribalCollege,
        DegreeOfUrbanization,
        InstitutionOpenToGeneralPublic,
        StatusOfInstitution,
        UnitidForMergedSchools,
        YearInstitutionWasDeletedFromIPEDS,
        DateInstitutionClosed,
        InstitutionIsActive,
        PrimarilyPostsecondaryIndicator,
        PostsecondaryInstitutionIndicator,
        PostsecondaryAndTitleIvInstitutionIndicator,
        ReportingMethodForStudentCharges,
        InstitutionalCategory,
        LandGrantInstitution,
        InstitutionSizeCategory,
```

```
        MultiCampusOrganization,
        NameOfMultiCampusOrganization,
        IdentificationNumberOfMultiCampusOrganization,
        CoreBasedStatisticalArea,
        CBSATypeMetropolitanMicropolitan,
        CombinedStatisticalArea,
        NewEnglandCityAndTownArea,
        FIPSCountyCode,
        CountyName,
        StateAnd114thCongressionalDistrictID,
        LongitudeLocation,
        LatitudeLocation,
        NCESGroupCategory,
        DataFeedbackReport,
        AcademicYear,
        CarnegieClassification,
        WebAddress,
        RecordCreateDateTime)
VALUES  (
        S.InstitutionIdentifier,
        S.InstitutionName,
        S.InstitutionNameAlias,
        S.StreetAddress,
        S.City,
        S.State,
        S.ZipCode,
        S.StateCode,
        S.EconomicAnalysisRegions,
        S.ChiefAdministrator,
        S.ChiefAdministratorTitle,
        S.TelephoneNumber,
        S.EmployerIdentificationNumber,
        S.DunBradstreetNumbers,
        S.PostsecondaryEducationIDNumber,
        S.TitleIVEligibilityIndicatorCode,
        S.SectorOfInstitution,
        S.LevelOfInstitution,
        S.ControlOfInstitution,
        S.HighestLevelOfOffering,
        S.UndergraduateOffering,
        S.GraduateOffering,
        S.HighestDegreeOffered,
        S.DegreeGrantingStatus,
        S.HistoricallyBlackCollegeOrUniversity,
        S.InstitutionHasHospital,
        S.InstitutionGrantsMedicalDegree,
        S.TribalCollege,
        S.DegreeOfUrbanization,
        S.InstitutionOpenToGeneralPublic,
        S.StatusOfInstitution,
        S.UnitidForMergedSchools,
        S.YearInstitutionWasDeletedFromIPEDS,
        S.DateInstitutionClosed,
        S.InstitutionIsActive,
        S.PrimarilyPostsecondaryIndicator,
```

```
        S.PostsecondaryInstitutionIndicator,
        S.PostsecondaryAndTitleIvInstitutionIndicator,
        S.ReportingMethodForStudentCharges,
        S.InstitutionalCategory,
        S.LandGrantInstitution,
        S.InstitutionSizeCategory,
        S.MultiCampusOrganization,
        S.NameOfMultiCampusOrganization,
        S.IdentificationNumberOfMultiCampusOrganization,
        S.CoreBasedStatisticalArea,
        S.CBSATypeMetropolitanMicropolitan,
        S.CombinedStatisticalArea,
        S.NewEnglandCityAndTownArea,
        S.FIPSCountyCode,
        S.CountyName,
        S.StateAnd114thCongressionalDistrictID,
        S.LongitudeLocation,
        S.LatitudeLocation,
        S.NCESGroupCategory,
        S.DataFeedbackReport,
        S.AcademicYear,
        OBJECT_CONSTRUCT (

    'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'CarnegieClassification2000' ,
                S.CarnegieClassification2000,
        'CarnegieClassification20052010Basic' ,
                S.CarnegieClassification20052010Basic,
        'CarnegieClassification2015Basic' ,
                S.CarnegieClassification2015Basic,
        'CarnegieClassification2015UndergraduateProgram' ,
                S.CarnegieClassification2015UndergraduateProgram,
        'CarnegieClassification2015GraduateProgram' ,
                S.CarnegieClassification2015GraduateProgram,
        'CarnegieClassification2015UndergraduateProfile' ,
                S.CarnegieClassification2015UndergraduateProfile,
        'CarnegieClassification2015EnrollmentProfile' ,
                S.CarnegieClassification2015EnrollmentProfile,
        'CarnegieClassification2015SizeSetting' ,
                S.CarnegieClassification2015SizeSetting ,
        'CarnegieClassification2018Basic',
                S.CarnegieClassification2018Basic,
        'CarnegieClassification2018UndergraduateProgram' ,
                S.CarnegieClassification2018UndergraduateProgram,
        'CarnegieClassification2018GraduateProgram',
                S.CarnegieClassification2018GraduateProgram,
        'CarnegieClassification2018UndergraduateProfile' ,
                S.CarnegieClassification2018UndergraduateProfile,
        'CarnegieClassification2018EnrollmentProfile',
                S.CarnegieClassification2018EnrollmentProfile,
        'CarnegieClassification2018SizeSetting' ,
                S.CarnegieClassification2018SizeSetting
          ) ,
        OBJECT_CONSTRUCT (
```

```
        'AcademicInstitutionIdentifier',
                S.InstitutionIdentifier,
        'InstitutionsWebAddress',
                S.InstitutionsWebAddress,
        'AdmissionsOfficeWebAddress',
                S.AdmissionsOfficeWebAddress,
        'FinancialAidOfficeWebAddress',
                S.FinancialAidOfficeWebAddress,
        'OnlineApplicationWebAddress',
                S.OnlineApplicationWebAddress,
        'NetPriceCalculatorWebAddress',
                S.NetPriceCalculatorWebAddress,
        'VeteransMilitaryServiceTuitionPoliciesWebAddress',
                S.VeteransMilitaryServiceTuitionPoliciesWebAddress,
        'StudentRightAthleteGraduationRateWebAddress',
                S.StudentRightAthleteGraduationRateWebAddress,
        'DisabilityServicesWebAddress',
                S.DisabilityServicesWebAddress
           ),
        CURRENT_TIMESTAMP
                );
`
try {
    snowflake.execute (
      {sqlText: sql_command}
      );
    return "Succeeded.";  // Return a success/error indicator.
    }
  catch (err) {
    return "Failed: " + err;  // Return a success/error indicator.
    }
  $$
  ;
```

8.10 Summary

In this chapter we have travelled through the process involved in loading and processing academic institution files for the years 2017, 2018, and 2019. We created a staged file and a corresponding operational data table to store institution details by year.We discussed in detail about the views and stored procedures used to populate the dimension table using the Snowflake MERGE command. We also demonstrated how to group various columns in the staged file using JSON string and VARIANT data types. Furthermore, we discussed the transformation logic to handle the schema drifts in the incoming data files. Finally, we demonstrated yet another approach to MERGE dimension table using hash keys.

> ***Note***
>
> Code available to download at
>
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/source/code/SQL/AcademicInstitution

9 JSON Ingestion

9.1 Enrollment

This section covers the information about enrollments in various institutions for the years 2017, 2018 and 2019. The enrollments are categorized by race, ethnicity and sex. The data files for enrollments are in the JSON format and the staged file is invoked using @IPEDS_EFFY.

We will learn about

- JSON ingestion
- Transforming data using the Snowflake JavaScripts stored procedures
- Calling stored procedures
-

```
SELECT
 S.$1:EFFYLEV::INTEGER  AS LevelOfStudent
,S.$1:EFY2MORM::INTEGER AS TwoOrMoreRacesMen
,S.$1:EFY2MORT::INTEGER AS TwoOrMoreRacesTotal
,S.$1:EFY2MORW::INTEGER AS TwoOrMoreRacesWomen
,S.$1:EFYAIANM::INTEGER AS AmericanIndianOrAlaskaNativeMen
,S.$1:EFYAIANT::INTEGER AS AmericanIndianOrAlaskaNativeTotal
,S.$1:EFYAIANW::INTEGER AS AmericanIndianOrAlaskaNativeWomen
,S.$1:EFYASIAM::INTEGER AS AsianMen
,S.$1:EFYASIAT::INTEGER AS AsianTotal
,S.$1:EFYASIAW::INTEGER AS AsianWomen
,S.$1:EFYBKAAM::INTEGER AS AfricanAmericanMen
,S.$1:EFYBKAAT::INTEGER AS AfricanAmericanTotal
,S.$1:EFYBKAAW::INTEGER AS AfricanAmericanWomen
,S.$1:EFYHISPM::INTEGER AS HispanicOrLatinoMen
,S.$1:EFYHISPT::INTEGER AS HispanicOrLatinoTotal
,S.$1:EFYHISPW::INTEGER AS HispanicOrLatinoWomen
,S.$1:EFYNHPIM::INTEGER AS HawaiianPacificIslanderMen
,S.$1:EFYNHPIT::INTEGER AS HawaiianPacificIslanderTotal
,S.$1:EFYNHPIW::INTEGER AS HawaiianPacificIslanderWomen
,S.$1:EFYNRALM::INTEGER AS NonresidentAlienMen
,S.$1:EFYNRALT::INTEGER AS NonresidentAlienTotal
,S.$1:EFYNRALW::INTEGER AS NonresidentAlienWomen
,S.$1:EFYTOTLM::INTEGER AS TotalMen
,S.$1:EFYTOTLT::INTEGER AS Total
,S.$1:EFYTOTLW::INTEGER AS TotalWomen
,S.$1:EFYUNKNM::INTEGER AS RaceEthnicityUnknownMen
,S.$1:EFYUNKNT::INTEGER AS RaceEthnicityUnknownTotal
,S.$1:EFYUNKNW::INTEGER AS RaceEthnicityUnknownWomen
,S.$1:EFYWHITM::INTEGER AS WhiteMen
,S.$1:EFYWHITT::INTEGER AS WhiteTotal
,S.$1:EFYWHITW::INTEGER AS WhiteWomen
,S.$1:LSTUDY::INTEGER AS OriginalLevelOfStudyOnSurveyForm
,S.$1:UNITID::INTEGER AS InstitutionIdentifier
,S.$1:XEFY2MOM::STRING AS ImputationVarTwoOrMoreRacesMen
,S.$1:XEFY2MOT::STRING AS ImputationVarTwoOrMoreRacesTotal
,S.$1:XEFY2MOW::STRING AS ImputationVarTwoOrMoreRacesWomen
,S.$1:XEFYAIAM::STRING AS ImputationVarAmericanIndianOrAlaskaNativeMen
```

```
,S.$1:XEFYAIAT::STRING AS ImputationVarAmericanIndianOrAlaskaNativeTotal
,S.$1:XEFYAIAW::STRING AS ImputationVarAmericanIndianOrAlaskaNativeWomen
,S.$1:XEFYASIM::STRING AS ImputationVarAsianMen
,S.$1:XEFYASIT::STRING AS ImputationVarAsianTotal
,S.$1:XEFYASIW::STRING AS ImputationVarAsianWomen
,S.$1:XEFYBKAM::STRING AS ImputationVarAfricanAmericanMen
,S.$1:XEFYBKAT::STRING AS ImputationVarAfricanAmericanTotal
,S.$1:XEFYBKAW::STRING AS ImputationVarAfricanAmericanWomen
,S.$1:XEFYHISM::STRING AS ImputationVarHispanicOrLatinoMen
,S.$1:XEFYHIST::STRING AS ImputationVarHispanicOrLatinoTotal
,S.$1:XEFYHISW::STRING AS ImputationVarHispanicOrLatinoWomen
,S.$1:XEFYNHPM::STRING AS ImputationVarHawaiianPacificIslanderMen
,S.$1:XEFYNHPT::STRING AS ImputationVarHawaiianPacificIslanderTotal
,S.$1:XEFYNHPW::STRING AS ImputationVarHawaiianPacificIslanderWomen
,S.$1:XEFYWHIM::STRING AS ImputationVarWhiteMen
,S.$1:XEFYWHIT::STRING AS ImputationVarWhiteTotal
,S.$1:XEFYWHIW::STRING AS ImputationVarWhiteWomen
,S.$1:XEYNRALM::STRING AS ImputationVarNonresidentAlienMen
,S.$1:XEYNRALT::STRING AS ImputationVarNonresidentAlienTotal
,S.$1:XEYNRALW::STRING AS ImputationVarNonresidentAlienWomen
,S.$1:XEYTOTLM::STRING AS ImputationVarTotalMen
,S.$1:XEYTOTLT::STRING AS ImputationVarTotal
,S.$1:XEYTOTLW::STRING AS ImputationVarTotalWomen
,S.$1:XEYUNKNM::STRING AS ImputationVarRaceEthnicityUnknownMen
,S.$1:XEYUNKNT::STRING AS ImputationVarRaceEthnicityUnknownTotal
,S.$1:XEYUNKNW::STRING AS ImputationVarRaceEthnicityUnknownWomen
FROM @IPEDS_EFFY S
WHERE RIGHT(REPLACE(S.metadata$filename,'_rv.json.gz',''),4) = 2017;
```

9.2 Operational data (OD) table for Enrollment

For storing the enrollments from the staged file, we have added a operational table called **od_Enrollment**. Here is the CREATE command for the staging table.

```
DROP TABLE IF EXISTS od_Enrollment;

CREATE TABLE od_Enrollment (
  LevelOfStudent INTEGER,
  TwoOrMoreRacesMen INTEGER,
  TwoOrMoreRacesTotal INTEGER,
  TwoOrMoreRacesWomen INTEGER,
  AmericanIndianOrAlaskaNativeMen INTEGER,
  AmericanIndianOrAlaskaNativeTotal INTEGER,
  AmericanIndianOrAlaskaNativeWomen INTEGER,
  AsianMen INTEGER,
  AsianTotal INTEGER,
  AsianWomen INTEGER,
  AfricanAmericanMen INTEGER,
  AfricanAmericanTotal INTEGER,
  AfricanAmericanWomen INTEGER,
  HispanicOrLatinoMen INTEGER,
  HispanicOrLatinoTotal INTEGER,
  HispanicOrLatinoWomen INTEGER,
  HawaiianPacificIslanderMen INTEGER,
```

```
  HawaiianPacificIslanderTotal INTEGER,
  HawaiianPacificIslanderWomen INTEGER,
  NonresidentAlienMen INTEGER,
  NonresidentAlienTotal INTEGER,
  NonresidentAlienWomen INTEGER,
  TotalMen INTEGER,
  Total INTEGER,
  TotalWomen INTEGER,
  RaceEthnicityUnknownMen INTEGER,
  RaceEthnicityUnknownTotal INTEGER,
  RaceEthnicityUnknownWomen INTEGER,
  WhiteMen INTEGER,
  WhiteTotal INTEGER,
  WhiteWomen INTEGER,
  OriginalLevelOfStudyOnSurveyForm INTEGER,
  InstitutionIdentifier INTEGER,
  ImputationVarTwoOrMoreRacesMen STRING,
  ImputationVarTwoOrMoreRacesTotal STRING,
  ImputationVarTwoOrMoreRacesWomen STRING,
  ImputationVarAmericanIndianOrAlaskaNativeMen STRING,
  ImputationVarAmericanIndianOrAlaskaNativeTotal STRING,
  ImputationVarAmericanIndianOrAlaskaNativeWomen STRING,
  ImputationVarAsianMen STRING,
  ImputationVarAsianTotal STRING,
  ImputationVarAsianWomen STRING,
  ImputationVarAfricanAmericanMen STRING,
  ImputationVarAfricanAmericanTotal STRING,
  ImputationVarAfricanAmericanWomen STRING,
  ImputationVarHispanicOrLatinoMen STRING,
  ImputationVarHispanicOrLatinoTotal STRING,
  ImputationVarHispanicOrLatinoWomen STRING,
  ImputationVarHawaiianPacificIslanderMen STRING,
  ImputationVarHawaiianPacificIslanderTotal STRING,
  ImputationVarHawaiianPacificIslanderWomen STRING,
  ImputationVarWhiteMen STRING,
  ImputationVarWhiteTotal STRING,
  ImputationVarWhiteWomen STRING,
  ImputationVarNonresidentAlienMen STRING,
  ImputationVarNonresidentAlienTotal STRING,
  ImputationVarNonresidentAlienWomen STRING,
  ImputationVarTotalMen STRING,
  ImputationVarTotal STRING,
  ImputationVarTotalWomen STRING,
  ImputationVarRaceEthnicityUnknownMen STRING,
  ImputationVarRaceEthnicityUnknownTotal STRING,
  ImputationVarRaceEthnicityUnknownWomen STRING,
  AcademicYear INTEGER,
  IngestedFileName STRING,
  RowNumber INTEGER
);
```

Once the operational data table is created, we will be populating the staging table with data from the source system, coming in the JSON file. We are using the stored procedures based ELT based approach for loading the OD tables. The stored procedure **pr_od_Enrollment_Load** used a parameter YEAR. Here is the Snowflake script for creating the stored procedure **pr_od_Enrollment_Load**.

```
CREATE OR REPLACE PROCEDURE
pr_od_Enrollment_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
AS
//Delete and Insert data to fact
$$
var sql_Command = `
INSERT INTO od_Enrollment (
LevelOfStudent,
TwoOrMoreRacesMen,
TwoOrMoreRacesTotal,
TwoOrMoreRacesWomen,
AmericanIndianOrAlaskaNativeMen,
AmericanIndianOrAlaskaNativeTotal,
AmericanIndianOrAlaskaNativeWomen,
AsianMen,
AsianTotal,
AsianWomen,
AfricanAmericanMen,
AfricanAmericanTotal,
AfricanAmericanWomen,
HispanicOrLatinoMen,
HispanicOrLatinoTotal,
HispanicOrLatinoWomen,
HawaiianPacificIslanderMen,
HawaiianPacificIslanderTotal,
HawaiianPacificIslanderWomen,
NonresidentAlienMen,
NonresidentAlienTotal,
NonresidentAlienWomen,
TotalMen,
Total,
TotalWomen,
RaceEthnicityUnknownMen,
RaceEthnicityUnknownTotal,
RaceEthnicityUnknownWomen,
WhiteMen,
WhiteTotal,
WhiteWomen,
OriginalLevelOfStudyOnSurveyForm,
InstitutionIdentifier,
ImputationVarTwoOrMoreRacesMen,
ImputationVarTwoOrMoreRacesTotal,
ImputationVarTwoOrMoreRacesWomen,
ImputationVarAmericanIndianOrAlaskaNativeMen,
ImputationVarAmericanIndianOrAlaskaNativeTotal,
ImputationVarAmericanIndianOrAlaskaNativeWomen,
ImputationVarAsianMen,
ImputationVarAsianTotal,
ImputationVarAsianWomen,
ImputationVarAfricanAmericanMen,
ImputationVarAfricanAmericanTotal,
ImputationVarAfricanAmericanWomen,
ImputationVarHispanicOrLatinoMen,
```

```
ImputationVarHispanicOrLatinoTotal,
ImputationVarHispanicOrLatinoWomen,
ImputationVarHawaiianPacificIslanderMen,
ImputationVarHawaiianPacificIslanderTotal,
ImputationVarHawaiianPacificIslanderWomen,
ImputationVarWhiteMen,
ImputationVarWhiteTotal,
ImputationVarWhiteWomen,
ImputationVarNonresidentAlienMen,
ImputationVarNonresidentAlienTotal,
ImputationVarNonresidentAlienWomen,
ImputationVarTotalMen,
ImputationVarTotal,
ImputationVarTotalWomen,
ImputationVarRaceEthnicityUnknownMen,
ImputationVarRaceEthnicityUnknownTotal,
ImputationVarRaceEthnicityUnknownWomen,
AcademicYear,
IngestedFileName,
RowNumber
)

SELECT
 S.$1:EFFYLEV::INTEGER  AS LevelOfStudent
,S.$1:EFY2MORM::INTEGER AS TwoOrMoreRacesMen
,S.$1:EFY2MORT::INTEGER AS TwoOrMoreRacesTotal
,S.$1:EFY2MORW::INTEGER AS TwoOrMoreRacesWomen
,S.$1:EFYAIANM::INTEGER AS AmericanIndianOrAlaskaNativeMen
,S.$1:EFYAIANT::INTEGER AS AmericanIndianOrAlaskaNativeTotal
,S.$1:EFYAIANW::INTEGER AS AmericanIndianOrAlaskaNativeWomen
,S.$1:EFYASIAM::INTEGER AS AsianMen
,S.$1:EFYASIAT::INTEGER AS AsianTotal
,S.$1:EFYASIAW::INTEGER AS AsianWomen
,S.$1:EFYBKAAM::INTEGER AS AfricanAmericanMen
,S.$1:EFYBKAAT::INTEGER AS AfricanAmericanTotal
,S.$1:EFYBKAAW::INTEGER AS AfricanAmericanWomen
,S.$1:EFYHISPM::INTEGER AS HispanicOrLatinoMen
,S.$1:EFYHISPT::INTEGER AS HispanicOrLatinoTotal
,S.$1:EFYHISPW::INTEGER AS HispanicOrLatinoWomen
,S.$1:EFYNHPIM::INTEGER AS HawaiianPacificIslanderMen
,S.$1:EFYNHPIT::INTEGER AS HawaiianPacificIslanderTotal
,S.$1:EFYNHPIW::INTEGER AS HawaiianPacificIslanderWomen
,S.$1:EFYNRALM::INTEGER AS NonresidentAlienMen
,S.$1:EFYNRALT::INTEGER AS NonresidentAlienTotal
,S.$1:EFYNRALW::INTEGER AS NonresidentAlienWomen
,S.$1:EFYTOTLM::INTEGER AS TotalMen
,S.$1:EFYTOTLT::INTEGER AS Total
,S.$1:EFYTOTLW::INTEGER AS TotalWomen
,S.$1:EFYUNKNM::INTEGER AS RaceEthnicityUnknownMen
,S.$1:EFYUNKNT::INTEGER AS RaceEthnicityUnknownTotal
,S.$1:EFYUNKNW::INTEGER AS RaceEthnicityUnknownWomen
,S.$1:EFYWHITM::INTEGER AS WhiteMen
,S.$1:EFYWHITT::INTEGER AS WhiteTotal
,S.$1:EFYWHITW::INTEGER AS WhiteWomen
,S.$1:LSTUDY::INTEGER AS OriginalLevelOfStudyOnSurveyForm
```

```
,S.$1:UNITID::INTEGER AS InstitutionIdentifier
,S.$1:XEFY2MOM::STRING AS ImputationVarTwoOrMoreRacesMen
,S.$1:XEFY2MOT::STRING AS ImputationVarTwoOrMoreRacesTotal
,S.$1:XEFY2MOW::STRING AS
                ImputationVarTwoOrMoreRacesWomen
,S.$1:XEFYAIAM::STRING AS
                ImputationVarAmericanIndianOrAlaskaNativeMen
,S.$1:XEFYAIAT::STRING AS
                ImputationVarAmericanIndianOrAlaskaNativeTotal
,S.$1:XEFYAIAW::STRING AS
                ImputationVarAmericanIndianOrAlaskaNativeWomen
,S.$1:XEFYASIM::STRING AS ImputationVarAsianMen
,S.$1:XEFYASIT::STRING AS ImputationVarAsianTotal
,S.$1:XEFYASIW::STRING AS ImputationVarAsianWomen
,S.$1:XEFYBKAM::STRING AS ImputationVarAfricanAmericanMen
,S.$1:XEFYBKAT::STRING AS ImputationVarAfricanAmericanTotal
,S.$1:XEFYBKAW::STRING AS ImputationVarAfricanAmericanWomen
,S.$1:XEFYHISM::STRING AS ImputationVarHispanicOrLatinoMen
,S.$1:XEFYHIST::STRING AS ImputationVarHispanicOrLatinoTotal
,S.$1:XEFYHISW::STRING AS ImputationVarHispanicOrLatinoWomen
,S.$1:XEFYNHPM::STRING AS ImputationVarHawaiianPacificIslanderMen
,S.$1:XEFYNHPT::STRING AS ImputationVarHawaiianPacificIslanderTotal
,S.$1:XEFYNHPW::STRING AS ImputationVarHawaiianPacificIslanderWomen
,S.$1:XEFYWHIM::STRING AS ImputationVarWhiteMen
,S.$1:XEFYWHIT::STRING AS ImputationVarWhiteTotal
,S.$1:XEFYWHIW::STRING AS ImputationVarWhiteWomen
,S.$1:XEYNRALM::STRING AS ImputationVarNonresidentAlienMen
,S.$1:XEYNRALT::STRING AS ImputationVarNonresidentAlienTotal
,S.$1:XEYNRALW::STRING AS ImputationVarNonresidentAlienWomen
,S.$1:XEYTOTLM::STRING AS ImputationVarTotalMen
,S.$1:XEYTOTLT::STRING AS ImputationVarTotal
,S.$1:XEYTOTLW::STRING AS ImputationVarTotalWomen
,S.$1:XEYUNKNM::STRING AS
                ImputationVarRaceEthnicityUnknownMen
,S.$1:XEYUNKNT::STRING AS
                ImputationVarRaceEthnicityUnknownTotal
,S.$1:XEYUNKNW::STRING AS
                ImputationVarRaceEthnicityUnknownWomen
,RIGHT(REPLACE(S.metadata$filename,'_rv.json.gz',''),4)::INTEGER
                AS AcademicYear
,S.metadata$filename::VARCHAR AS IngestedFileName
,S.metadata$file_row_number::INTEGER AS RowNumber
FROM @IPEDS_EFFY S
WHERE RIGHT(REPLACE(S.metadata$filename,'_rv.json.gz',''),4)
= `+ YEAR.toString()

try
{
snowflake.execute(
{sqlText: sql_Command}
);
}
catch(err)
{
return "Failed :" + err;
```

```
}
return "success"
$$;
```

We can load data for all these 3 years. Here are the commands to call the stored procedure.

```
TRUNCATE TABLE od_Enrollment;

CALL pr_od_Enrollment_Load(2017::FLOAT);
CALL pr_od_Enrollment_Load(2018::FLOAT);
CALL pr_od_Enrollment_Load(2019::FLOAT);
```

9.3 Enrollment load

We have created a table **Enrollment** in the servicing or consuming layer. Here is the script for creating the table Enrollment.

```
CREATE OR REPLACE TABLE Enrollment (
  AcademicInstitutionUniqueDWSID INTEGER,
  InstitutionIdentifier INTEGER,
  AcademicYear INTEGER,
  LevelOfStudent INTEGER,
  OriginalLevelOfStudyOnSurveyForm INTEGER,
  TotalMen INTEGER,
  TotalWomen INTEGER,
  WhiteMen INTEGER,
  WhiteWomen INTEGER,
  AfricanAmericanMen INTEGER,
  AfricanAmericanWomen INTEGER,
  AmericanIndianOrAlaskaNativeMen INTEGER,
  AmericanIndianOrAlaskaNativeWomen INTEGER,
  AsianMen INTEGER,
  AsianWomen INTEGER,
  HispanicOrLatinoMen INTEGER,
  HispanicOrLatinoWomen INTEGER,
  HawaiianPacificIslanderMen INTEGER,
  HawaiianPacificIslanderWomen INTEGER,
  NonresidentAlienMen INTEGER,
  NonresidentAlienWomen INTEGER,
  TwoOrMoreRacesMen INTEGER,
  TwoOrMoreRacesWomen INTEGER,
  RaceEthnicityUnknownMen INTEGER,
  RaceEthnicityUnknownWomen INTEGER,
  ImputationVarTotalMen VARCHAR(10),
  ImputationVarTotalWomen VARCHAR(10),
  ImputationVarWhiteMen VARCHAR(10),
  ImputationVarWhiteWomen VARCHAR(10),
  ImputationVarAfricanAmericanMen VARCHAR(10),
  ImputationVarAfricanAmericanWomen VARCHAR(10),
  ImputationVarAmericanIndianOrAlaskaNativeMen VARCHAR(10),
  ImputationVarAmericanIndianOrAlaskaNativeWomen VARCHAR(10),
  ImputationVarHispanicOrLatinoMen VARCHAR(10),
  ImputationVarHispanicOrLatinoWomen VARCHAR(10),
  ImputationVarAsianMen VARCHAR(10),
```

```
  ImputationVarAsianWomen VARCHAR(10),
  ImputationVarHawaiianPacificIslanderMen VARCHAR(10),
  ImputationVarHawaiianPacificIslanderWomen VARCHAR(10),
  ImputationVarNonresidentAlienMen VARCHAR(10),
  ImputationVarNonresidentAlienWomen VARCHAR(10),
  ImputationVarTwoOrMoreRacesMen VARCHAR(10),
  ImputationVarTwoOrMoreRacesWomen VARCHAR(10),
  ImputationVarRaceEthnicityUnknownMen VARCHAR(10),
  ImputationVarRaceEthnicityUnknownWomen VARCHAR(10),
  VariablesImputedCount INTEGER,
  VariablesReportedCount INTEGER,
  ImputationVarPercent NUMERIC(10,4),
  ReportedVarPercent NUMERIC(10,4)
);
```

The stored procedure **pr_Enrollment_Load** is used for loading this servicing layer table. The stored procedure loads data for specific years based on the parameter passed in to it. Each time this stored procedure is called, it deletes the records from the Enrollment table for that year and re-inserts them from od_Enrollment for the year.

While loading data into the servicing layer load ,data transformation is added. The imputation variable determines whether the value in the numeric field is imputed or reported. We are interested to understand the total number of imputed variable counts, and reported variable counts. Those values are stored in **VariablesReportedCount** and **VariablesImputedCount**. Based on the values of VariablesReportedCount and VariablesImputedCount, the percentage of imputed or reported is calculated. Those percentages are stored in **ImputationVarPercent**, and **ReportedVarPercent**

Here is the script for stored procedure **pr_Enrollment_Load**. The additional data transformation needed to get the above variables, are computed in the same SP.

```
CREATE
        OR REPLACE PROCEDURE pr_Enrollment_Load (YEAR FLOAT)
RETURNS STRING LANGUAGE javascript
                AS $$ var sql_DelCommand = `DELETE
FROM Enrollment
WHERE AcademicYear = ` + YEAR.toString() + `;

` var sql_Command = `

INSERT INTO Enrollment (
        AcademicInstitutionUniqueDWSID
        ,InstitutionIdentifier
        ,AcademicYear
        ,LevelOfStudent
        ,OriginalLevelOfStudyOnSurveyForm
        ,TotalMen
        ,TotalWomen
        ,WhiteMen
        ,WhiteWomen
        ,AfricanAmericanMen
        ,AfricanAmericanWomen
        ,AmericanIndianOrAlaskaNativeMen
        ,AmericanIndianOrAlaskaNativeWomen
        ,AsianMen
        ,AsianWomen
```

```
        ,HispanicOrLatinoMen
        ,HispanicOrLatinoWomen
        ,HawaiianPacificIslanderMen
        ,HawaiianPacificIslanderWomen
        ,NonresidentAlienMen
        ,NonresidentAlienWomen
        ,TwoOrMoreRacesMen
        ,TwoOrMoreRacesWomen
        ,RaceEthnicityUnknownMen
        ,RaceEthnicityUnknownWomen
        ,ImputationVarTotalMen
        ,ImputationVarTotalWomen
        ,ImputationVarWhiteMen
        ,ImputationVarWhiteWomen
        ,ImputationVarAfricanAmericanMen
        ,ImputationVarAfricanAmericanWomen
        ,ImputationVarAmericanIndianOrAlaskaNativeMen
        ,ImputationVarAmericanIndianOrAlaskaNativeWomen
        ,ImputationVarHispanicOrLatinoMen
        ,ImputationVarHispanicOrLatinoWomen
        ,ImputationVarAsianMen
        ,ImputationVarAsianWomen
        ,ImputationVarHawaiianPacificIslanderMen
        ,ImputationVarHawaiianPacificIslanderWomen
        ,ImputationVarNonresidentAlienMen
        ,ImputationVarNonresidentAlienWomen
        ,ImputationVarTwoOrMoreRacesMen
        ,ImputationVarTwoOrMoreRacesWomen
        ,ImputationVarRaceEthnicityUnknownMen
        ,ImputationVarRaceEthnicityUnknownWomen
        ,VariablesImputedCount
        ,VariablesReportedCount
        )
SELECT IFNULL(D.AcademicInstitutionUniqueDWSID, - 1)
        ,S.InstitutionIdentifier
        ,S.AcademicYear
        ,S.LevelOfStudent
        ,S.OriginalLevelOfStudyOnSurveyForm
        ,S.TotalMen
        ,S.TotalWomen
        ,S.WhiteMen
        ,S.WhiteWomen
        ,S.AfricanAmericanMen
        ,S.AfricanAmericanWomen
        ,S.AmericanIndianOrAlaskaNativeMen
        ,S.AmericanIndianOrAlaskaNativeWomen
        ,S.AsianMen
        ,S.AsianWomen
        ,S.HispanicOrLatinoMen
        ,S.HispanicOrLatinoWomen
        ,S.HawaiianPacificIslanderMen
        ,S.HawaiianPacificIslanderWomen
        ,S.NonresidentAlienMen
        ,S.NonresidentAlienWomen
        ,S.TwoOrMoreRacesMen
```

```
        ,S.TwoOrMoreRacesWomen
        ,S.RaceEthnicityUnknownMen
        ,S.RaceEthnicityUnknownWomen
        ,S.ImputationVarTotalMen
        ,S.ImputationVarTotalWomen
        ,S.ImputationVarWhiteMen
        ,S.ImputationVarWhiteWomen
        ,S.ImputationVarAfricanAmericanMen
        ,S.ImputationVarAfricanAmericanWomen
        ,S.ImputationVarAmericanIndianOrAlaskaNativeMen
        ,S.ImputationVarAmericanIndianOrAlaskaNativeWomen
        ,S.ImputationVarAsianMen
        ,S.ImputationVarAsianWomen
        ,S.ImputationVarHispanicOrLatinoMen
        ,S.ImputationVarHispanicOrLatinoWomen
        ,S.ImputationVarHawaiianPacificIslanderMen
        ,S.ImputationVarHawaiianPacificIslanderWomen
        ,S.ImputationVarNonresidentAlienMen
        ,S.ImputationVarNonresidentAlienWomen
        ,S.ImputationVarTwoOrMoreRacesMen
        ,S.ImputationVarTwoOrMoreRacesWomen
        ,S.ImputationVarRaceEthnicityUnknownMen
        ,S.ImputationVarRaceEthnicityUnknownWomen
        ,
        --- calculation
        (
        CASE
                WHEN S.ImputationVarTotalMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarTotalWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarWhiteMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarWhiteWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAfricanAmericanMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAfricanAmericanWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAmericanIndianOrAlaskaNativeMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
```

```
                WHEN S.ImputationVarAmericanIndianOrAlaskaNativeWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAsianMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAsianWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarHispanicOrLatinoMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarHispanicOrLatinoWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarNonresidentAlienMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarNonresidentAlienWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarHawaiianPacificIslanderMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarHawaiianPacificIslanderWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarTwoOrMoreRacesMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarTwoOrMoreRacesWomen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarRaceEthnicityUnknownMen <> 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarRaceEthnicityUnknownWomen <> 'R'
                        THEN 1
                ELSE 0
                END
        ) AS VariablesImputedCount
        ,(
        CASE
```

```
                WHEN S.ImputationVarTotalMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarTotalWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarWhiteMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarWhiteWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAfricanAmericanMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAfricanAmericanWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAmericanIndianOrAlaskaNativeMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAmericanIndianOrAlaskaNativeWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAsianMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarAsianWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarHispanicOrLatinoMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarHispanicOrLatinoWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarNonresidentAlienMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarNonresidentAlienWomen = 'R'
                        THEN 1
                ELSE 0
```

```
                END + CASE
                WHEN S.ImputationVarHawaiianPacificIslanderMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarHawaiianPacificIslanderWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarTwoOrMoreRacesMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarTwoOrMoreRacesWomen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarRaceEthnicityUnknownMen = 'R'
                        THEN 1
                ELSE 0
                END + CASE
                WHEN S.ImputationVarRaceEthnicityUnknownWomen = 'R'
                        THEN 1
                ELSE 0
                END
        ) AS VariablesReportedCount
FROM od_Enrollment S
INNER JOIN AcademicInstitution D ON S.InstitutionIdentifier = D.
        InstitutionIdentifier
WHERE S.ACADEMICYEAR = ` + YEAR.toString() var sql_UpdCommand = `

UPDATE Enrollment
SET ImputationVarPercent = VariablesImputedCount
        / (VariablesImputedCount + VariablesReportedCount
                )
        ,ReportedVarPercent = VariablesReportedCount
                / (VariablesImputedCount + VariablesReportedCount
                )
WHERE ACADEMICYEAR = ` + YEAR.toString() var arrSqlCmd = [3];

arrSqlCmd [0] = sql_DelCommand arrSqlCmd [1] = sql_Command arrSqlCmd [2] =
        sql_UpdCommand var sql_ExecCommand = '';
FOR (var j = 0;j < arrSqlCmd.length;j + +)

{ sql_ExecCommand = arrSqlCmd [j] try { snowflake.

EXECUTE ({sqlText: sql_ExecCommand});

} catch(err) {

RETURN "Failed :" + err;

} }

RETURN "success" $$;
```

The above SP is called to process data for 2017,2018, and 2019. The code below shows the call to the 06_CR_Enrollment_Load for loading all historical files.

```
TRUNCATE TABLE Enrollment;
CALL pr_Enrollment_Load(2017::FLOAT);
CALL pr_Enrollment_Load(2018::FLOAT);
CALL pr_Enrollment_Load(2019::FLOAT);
```

9.4 Summary

In this chapter we have discussed the enrollments in various institutions by year. We talked about creating a OD table from data files which are in a JSON format. We also discussed the servicing layer table and the (ETL code) stored procedure to load the servicing layer from the staging table.

> ***Note***
>
> Code available to download at
>
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/source/code/SQL/Enrollment

10 ORC Ingestion

10.1 Institutional Charges

This section covers the data ingestion about tuition fees charged by the numerous institutions for various courses and segment of students admitted during 2017, 2018 and 2019 academic years. The data files received on institutional charges are in ORC format. Dataset also categorize fees for graduate programs like Medicine, Dentistry, Pharmacy, Law etc. as different attributes within academic institution charge records.

Fees are also categorized for In-State and Out-of-State students. Books/supply , campus room, publication charges as well as other expenses are also available here.

The information in this large file is broadly categorized into 4 groups described below.

10.1.1 Institutional Charge (General)

This cross section of attributes cover average tuition/required fees, per-credit fees charged by educational instituites. Fees varies significantly from state to state, public vs private, and ranking of instituites (demand vs supply).

10.1.2 Institutional Charge by academic branch

The details of fees by academic subjects like Chiropractic, Dentistry, Medicine, Optometry, OsteopathicMedicine, Pharmacy, Podiatry, VeterinaryMedicine, and Law by various institutions for in-state and out-of-state students are available in this section.

10.1.3 Institutional Charge by residency type

The residency type determines different fees structure in public uninversities for the same courses. Normally in-state students are charged less than out-of-state. These set of attributes will provide insight around those variations.

10.1.4 Institutional Charge by category

Various fees like On-Campus boarding, books/supplies, and other on-campus expenses are available in this section of the file.

We will learn how to

- Use ingested ORC files from the staged Snowflake region
- Transform data using the Snowflake JavaScript stored procedures
- Call the stored procedures
- Perform transformation using Unpivot and JSON conversation

10.2 Staged File for Institution Charges

The staged file **@IPEDS_IC** is used for keeping track of various institution charges. We can retrieve the details of a particular year as below. Here is the Snowflake SQL query to select all the rows for the year 2017 from staged ORC data. Please note that the columns are selected using **$1 :<column_name>**. This is because data files used are in ORC format. The staged file is a denormalized data structure, where different attributes are added to represent differences in these fee structures. While loading data to the final servicing layer, data architects may consider to normlize these attributes in a meaningful way. In this book, we have followed some normalization while data is loaded into the servicing layer.

```
SELECT
$1:UNITID :: VARCHAR AS InstitutionIdentifier,
$1:TUITION1 :: INTEGER AS
    InDistrictAverageTuitionFullTimeUndergraduates,
$1:FEE1  :: INTEGER AS
    InDistrictRequiredFeesFullTimeUndergraduates,
$1:HRCHG1 :: INTEGER AS
    InDistrictPerCreditHourChargeForPartTimeUndergraduates,
$1:TUITION2 :: INTEGER AS InStateAverageTuitionFullTimeUndergraduates,
$1:FEE2  :: INTEGER AS
    InStateRequiredFeesFullTimeUndergraduates,
$1:HRCHG2 :: INTEGER AS
    InStatePerCreditHourChargeForPartTimeUndergraduates,
$1:TUITION3 :: INTEGER AS
    OutOfStateAverageTuitionFullTimeUndergraduates,
$1:FEE3 :: INTEGER AS
    OutOfStateRequiredFeesFullTimeUndergraduates,
$1:HRCHG3 :: INTEGER AS
    OutOfStatePerCreditHourChargeForPartTimeUndergraduates,
$1:TUITION5 :: INTEGER AS InDistrictAverageTuitionFullTimeGraduates,
$1:FEE5  :: INTEGER AS InDistrictRequiredFeesFullTimeGraduates,
$1:HRCHG5 :: INTEGER AS InDistrictPerCreditHourChargePartTimeGraduates,
$1:TUITION6 :: INTEGER AS InStateAverageTuitionFullTimeGraduates,
$1:FEE6  :: INTEGER AS InStateRequiredFeesFullTimeGraduates,
$1:HRCHG6 :: INTEGER AS InStatePerCreditHourChargePartTimeGraduates,
$1:TUITION7 :: INTEGER AS OutOfStateAverageTuitionFullTimeGraduates,
$1:FEE7  :: INTEGER AS OutOfStateRequiredFeesFullTimeGraduates,
$1:HRCHG7 :: INTEGER AS OutOfStatePerCreditHourChargePartTimeGraduate,
$1:ISPROF1 :: INTEGER AS Chiropractic_InStateTuition,
$1:ISPFEE1 :: INTEGER AS Chiropractic_InStateRequiredFees,
$1:OSPROF1 :: INTEGER AS Chiropractic_OutOfStateTuition,
$1:OSPFEE1 :: INTEGER AS Chiropractic_OutOfStateRequiredFees,
$1:ISPROF2 :: INTEGER AS Dentistry_InStateTuition,
$1:ISPFEE2 :: INTEGER AS Dentistry_InStateRequiredFees,
$1:OSPROF2 :: INTEGER AS Dentistry_OutOfStateTuition,
$1:OSPFEE2 :: INTEGER AS Dentistry_OutOfStateRequiredFees,
$1:ISPROF3 :: INTEGER AS Medicine_InStateTuition,
$1:ISPFEE3 :: INTEGER AS Medicine_InStateRequiredFees,
$1:OSPROF3 :: INTEGER AS Medicine_OutOfStateTuition,
$1:OSPFEE3 :: INTEGER AS Medicine_OutOfStateRequiredFees,
$1:ISPROF4 :: INTEGER AS Optometry_InStateTuition,
$1:ISPFEE4 :: INTEGER AS Optometry_InStateRequiredFees,
$1:OSPROF4 :: INTEGER AS Optometry_OutOfStateTuition,
$1:OSPFEE4 :: INTEGER AS Optometry_OutOfStateRequiredFees,
$1:ISPROF5 :: INTEGER AS OsteopathicMedicine_InStateTuition,
$1:ISPFEE5 :: INTEGER AS OsteopathicMedicine_InStateRequiredFees,
$1:OSPROF5 :: INTEGER AS OsteopathicMedicine_OutOfStateTuition,
$1:OSPFEE5 :: INTEGER AS OsteopathicMedicine_OutOfStateRequiredFees,
$1:ISPROF6 :: INTEGER AS Pharmacy_InStateTuition,
$1:ISPFEE6 :: INTEGER AS Pharmacy_InStateRequiredFees,
$1:OSPROF6 :: INTEGER AS Pharmacy_OutOfStateTuition,
$1:OSPFEE6 :: INTEGER AS Pharmacy_OutOfStateRequiredFees,
$1:ISPROF7 :: INTEGER AS Podiatry_InStateTuition,
$1:ISPFEE7 :: INTEGER AS Podiatry_InStateRequiredFees,
$1:OSPROF7 :: INTEGER AS Podiatry_OutOfStateTuition,
```

```
$1:OSPFEE7 :: INTEGER AS Podiatry_OutOfStateRequiredFees,
$1:ISPROF8 :: INTEGER AS VeterinaryMedicine_InStateTuition,
$1:ISPFEE8 :: INTEGER AS VeterinaryMedicine_InStateRequiredFees,
$1:OSPROF8 :: INTEGER AS VeterinaryMedicine_OutOfStateTuition,
$1:OSPFEE8 :: INTEGER AS VeterinaryMedicine_OutOfStateRequiredFees,
$1:ISPROF9 :: INTEGER AS Law_InStateTuition,
$1:ISPFEE9 :: INTEGER AS Law_InStateRequiredFees,
$1:OSPROF9 :: INTEGER AS Law_OutOfStateTuition,
$1:OSPFEE9 :: INTEGER AS Law_OutOfStateRequiredFees,
$1:CHG1AT0 :: INTEGER AS PublishedInDistrictTuitionYear0,
$1:CHG1AF0 :: INTEGER AS PublishedInDistrictFeesYear0,
$1:CHG1AY0 :: INTEGER AS PublishedInDistrictTuitionFeesYear0,
$1:CHG1AT1 :: INTEGER AS PublishedInDistrictTuitionYear1,
$1:CHG1AF1 :: INTEGER AS PublishedInDistrictFeesYear1,
$1:CHG1AY1 :: INTEGER AS PublishedInDistrictTuitionFeesYear1,
$1:CHG1AT2 :: INTEGER AS PublishedInDistrictTuitionYear2,
$1:CHG1AF2 :: INTEGER AS PublishedInDistrictFeesYear2,
$1:CHG1AY2 :: INTEGER AS PublishedInDistrictTuitionFeesYear2,
$1:CHG1AT3 :: INTEGER AS PublishedInDistrictTuitionYear3,
$1:CHG1AF3 :: INTEGER AS PublishedInDistrictFeesYear3,
$1:CHG1AY3 :: INTEGER AS PublishedInDistrictTuitionFeesYear3,
$1:CHG1TGTD :: INTEGER AS
    PublishedInDistrictTuitionYear3GuaranteedPercentIncrease,
$1:CHG1FGTD :: INTEGER AS
    PublishedInDistrictFeesYear3GuaranteedPercentIncrease,
$1:CHG2AT0 :: INTEGER AS PublishedInStateTuitionYear0,
$1:CHG2AF0 :: INTEGER AS PublishedInStateFeesYear0,
$1:CHG2AY0 :: INTEGER AS PublishedInStateTuitionFeesYear0,
$1:CHG2AT1 :: INTEGER AS PublishedInStateTuitionYear1,
$1:CHG2AF1 :: INTEGER AS PublishedInStateFeesYear1,
$1:CHG2AY1 :: INTEGER AS PublishedInStateTuitionFeesYear1,
$1:CHG2AT2 :: INTEGER AS PublishedInStateTuitionYear2,
$1:CHG2AF2 :: INTEGER AS PublishedInStateFeesYear2,
$1:CHG2AY2 :: INTEGER AS PublishedInStateTuitionFeesYear2,
$1:CHG2AT3 :: INTEGER AS PublishedInStateTuitionYear3,
$1:CHG2AF3 :: INTEGER AS PublishedInStateFeesYear3,
$1:CHG2AY3 :: INTEGER AS PublishedInStateTuitionFeesYear3,
$1:CHG2TGTD :: INTEGER AS PublishedInStateTuitionGuaranteedPercentIncrease,
$1:CHG2FGTD :: INTEGER AS PublishedInStateFeesGuaranteedPercentIncrease,
$1:CHG3AT0 :: INTEGER AS PublishedOutOfStateTuitionYear0,
$1:CHG3AF0 :: INTEGER AS PublishedOutOfStateFeesYear0,
$1:CHG3AY0 :: INTEGER AS PublishedOutOfStateTuitionFeesYear0,
$1:CHG3AT1 :: INTEGER AS PublishedOutOfStateTuitionYear1,
$1:CHG3AF1 :: INTEGER AS PublishedOutOfStateFeesYear1,
$1:CHG3AY1 :: INTEGER AS PublishedOutOfStateTuitionFeesYear1,
$1:CHG3AT2 :: INTEGER AS PublishedOutOfStateTuitionYear2,
$1:CHG3AF2 :: INTEGER AS PublishedOutOfStateFeesYear2,
$1:CHG3AY2 :: INTEGER AS PublishedOutOfStateTuitionFeesYear2,
$1:CHG3AT3 :: INTEGER AS PublishedOutOfStateTuitionYear3,
$1:CHG3AF3 :: INTEGER AS PublishedOutOfStateFeesYear3,
$1:CHG3AY3 :: INTEGER AS PublishedOutOfStateTuitionFeesYear3,
$1:CHG3TGTD :: INTEGER AS
    PublishedOutOfStateTuitionGuaranteedPercentIncrease,
$1:CHG3FGTD :: INTEGER AS
    PublishedOutOfStateFeesGuaranteedPercentIncrease,
```

```
$1:CHG4AY0 :: INTEGER AS BooksSuppliesYear0,
$1:CHG4AY1 :: INTEGER AS BooksSuppliesYear1,
$1:CHG4AY2 :: INTEGER AS BooksSuppliesYear2,
$1:CHG4AY3 :: INTEGER AS BooksSuppliesYear3,
$1:CHG5AY0 :: INTEGER AS OnCampusRoomBoardYear0,
$1:CHG5AY1 :: INTEGER AS OnCampusRoomBoardYear1,
$1:CHG5AY2 :: INTEGER AS OnCampusRoomBoardYear2,
$1:CHG5AY3 :: INTEGER AS OnCampusRoomBoardYear3,
$1:CHG6AY0 :: INTEGER AS OnCampusOtherExpensesYear0,
$1:CHG6AY1 :: INTEGER AS OnCampusOtherExpensesYear1,
$1:CHG6AY2 :: INTEGER AS OnCampusOtherExpensesYear2,
$1:CHG6AY3 :: INTEGER AS OnCampusOtherExpensesYear3,
$1:CHG7AY0 :: INTEGER AS OffCampusNotWithFamilyRoomBoardYear0,
$1:CHG7AY1 :: INTEGER AS OffCampusNotWithFamilyRoomBoardYear1,
$1:CHG7AY2 :: INTEGER AS OffCampusNotWithFamilyRoomBoardYear2,
$1:CHG7AY3 :: INTEGER AS OffCampusNotWithFamilyRoomBoardYear3,
$1:CHG8AY0 :: INTEGER AS OffCampusNotWithFamilyOtherExpensesYear0,
$1:CHG8AY1 :: INTEGER AS OffCampusNotWithFamilyOtherExpensesYear1,
$1:CHG8AY2 :: INTEGER AS OffCampusNotWithFamilyOtherExpensesYear2,
$1:CHG8AY3 :: INTEGER AS OffCampusNotWithFamilyOtherExpensesYear3,
$1:CHG9AY0 :: INTEGER AS OffCampusWithFamilyOtherExpensesYear0,
$1:CHG9AY1 :: INTEGER AS OffCampusWithFamilyOtherExpensesYear1,
$1:CHG9AY2 :: INTEGER AS OffCampusWithFamilyOtherExpensesYear2,
$1:CHG9AY3 :: INTEGER AS OffCampusWithFamilyOtherExpensesYear3,
LEFT(MetaData$FileName, 4) AS AcademicYear,
metadata$filename IngestedFileName,
metadata$file_row_number RowNumber
FROM
@IPEDS_IC
WHERE LEFT(MetaData$FileName,4) = 2017;
```

10.3 Operational data (OD) table for Institutional charge

OD table od_InstitutionalCharge is added to load institutional charge data from the staged file. OD table contains all the columns from the staged file @IPEDS_IC. The Snowflake script for creating the OD table is shown below.

```
CREATE OR REPLACE TABLE od_InstitutionalCharge (
  InstitutionIdentifier INTEGER,
  InDistrictAverageTuitionFullTimeUndergraduates INTEGER,
  InDistrictRequiredFeesFullTimeUndergraduates INTEGER,
  InDistrictPerCreditHourChargeForPartTimeUndergraduates INTEGER,
  InStateAverageTuitionFullTimeUndergraduates INTEGER,
  InStateRequiredFeesFullTimeUndergraduates INTEGER,
  InStatePerCreditHourChargeForPartTimeUndergraduates INTEGER,
  OutOfStateAverageTuitionFullTimeUndergraduates INTEGER,
  OutOfStateRequiredFeesFullTimeUndergraduates INTEGER,
  OutOfStatePerCreditHourChargeForPartTimeUndergraduates INTEGER,
  InDistrictAverageTuitionFullTimeGraduates INTEGER,
  InDistrictRequiredFeesFullTimeGraduates INTEGER,
  InDistrictPerCreditHourChargePartTimeGraduates INTEGER,
  InStateAverageTuitionFullTimeGraduates INTEGER,
  InStateRequiredFeesFullTimeGraduates INTEGER,
  InStatePerCreditHourChargePartTimeGraduates INTEGER,
```

```
OutOfStateAverageTuitionFullTimeGraduates INTEGER,
OutOfStateRequiredFeesFullTimeGraduates INTEGER,
OutOfStatePerCreditHourChargePartTimeGraduate INTEGER,
Chiropractic_InStateTuition INTEGER,
Chiropractic_InStateRequiredFees INTEGER,
Chiropractic_OutOfStateTuition INTEGER,
Chiropractic_OutOfStateRequiredFees INTEGER,
Dentistry_InStateTuition INTEGER,
Dentistry_InStateRequiredFees INTEGER,
Dentistry_OutOfStateTuition INTEGER,
Dentistry_OutOfStateRequiredFees INTEGER,
Medicine_InStateTuition INTEGER,
Medicine_InStateRequiredFees INTEGER,
Medicine_OutOfStateTuition INTEGER,
Medicine_OutOfStateRequiredFees INTEGER,
Optometry_InStateTuition INTEGER,
Optometry_InStateRequiredFees INTEGER,
Optometry_OutOfStateTuition INTEGER,
Optometry_OutOfStateRequiredFees INTEGER,
OsteopathicMedicine_InStateTuition INTEGER,
OsteopathicMedicine_InStateRequiredFees INTEGER,
OsteopathicMedicine_OutOfStateTuition INTEGER,
OsteopathicMedicine_OutOfStateRequiredFees INTEGER,
Pharmacy_InStateTuition INTEGER,
Pharmacy_InStateRequiredFees INTEGER,
Pharmacy_OutOfStateTuition INTEGER,
Pharmacy_OutOfStateRequiredFees INTEGER,
Podiatry_InStateTuition INTEGER,
Podiatry_InStateRequiredFees INTEGER,
Podiatry_OutOfStateTuition INTEGER,
Podiatry_OutOfStateRequiredFees INTEGER,
VeterinaryMedicine_InStateTuition INTEGER,
VeterinaryMedicine_InStateRequiredFees INTEGER,
VeterinaryMedicine_OutOfStateTuition INTEGER,
VeterinaryMedicine_OutOfStateRequiredFees INTEGER,
Law_InStateTuition INTEGER,
Law_InStateRequiredFees INTEGER,
Law_OutOfStateTuition INTEGER,
Law_OutOfStateRequiredFees INTEGER,
PublishedInDistrictTuitionYear0 INTEGER,
PublishedInDistrictFeesYear0 INTEGER,
PublishedInDistrictTuitionFeesYear0 INTEGER,
PublishedInDistrictTuitionYear1 INTEGER,
PublishedInDistrictFeesYear1 INTEGER,
PublishedInDistrictTuitionFeesYear1 INTEGER,
PublishedInDistrictTuitionYear2 INTEGER,
PublishedInDistrictFeesYear2 INTEGER,
PublishedInDistrictTuitionFeesYear2 INTEGER,
PublishedInDistrictTuitionYear3 INTEGER,
PublishedInDistrictFeesYear3 INTEGER,
PublishedInDistrictTuitionFeesYear3 INTEGER,
PublishedInDistrictTuitionYear3GuaranteedPercentIncrease INTEGER,
PublishedInDistrictFeesYear3GuaranteedPercentIncrease INTEGER,
PublishedInStateTuitionYear0 INTEGER,
PublishedInStateFeesYear0 INTEGER,
```

```
  PublishedInStateTuitionFeesYear0 INTEGER,
  PublishedInStateTuitionYear1 INTEGER,
  PublishedInStateFeesYear1 INTEGER,
  PublishedInStateTuitionFeesYear1 INTEGER,
  PublishedInStateTuitionYear2 INTEGER,
  PublishedInStateFeesYear2 INTEGER,
  PublishedInStateTuitionFeesYear2 INTEGER,
  PublishedInStateTuitionYear3 INTEGER,
  PublishedInStateFeesYear3 INTEGER,
  PublishedInStateTuitionFeesYear3 INTEGER,
  PublishedInStateTuitionGuaranteedPercentIncrease INTEGER,
  PublishedInStateFeesGuaranteedPercentIncrease INTEGER,
  PublishedOutOfStateTuitionYear0 INTEGER,
  PublishedOutOfStateFeesYear0 INTEGER,
  PublishedOutOfStateTuitionFeesYear0 INTEGER,
  PublishedOutOfStateTuitionYear1 INTEGER,
  PublishedOutOfStateFeesYear1 INTEGER,
  PublishedOutOfStateTuitionFeesYear1 INTEGER,
  PublishedOutOfStateTuitionYear2 INTEGER,
  PublishedOutOfStateFeesYear2 INTEGER,
  PublishedOutOfStateTuitionFeesYear2 INTEGER,
  PublishedOutOfStateTuitionYear3 INTEGER,
  PublishedOutOfStateFeesYear3 INTEGER,
  PublishedOutOfStateTuitionFeesYear3 INTEGER,
  PublishedOutOfStateTuitionGuaranteedPercentIncrease INTEGER,
  PublishedOutOfStateFeesGuaranteedPercentIncrease INTEGER,
  BooksSuppliesYear0 INTEGER,
  BooksSuppliesYear1 INTEGER,
  BooksSuppliesYear2 INTEGER,
  BooksSuppliesYear3 INTEGER,
  OnCampusRoomBoardYear0 INTEGER,
  OnCampusRoomBoardYear1 INTEGER,
  OnCampusRoomBoardYear2 INTEGER,
  OnCampusRoomBoardYear3 INTEGER,
  OnCampusOtherExpensesYear0 INTEGER,
  OnCampusOtherExpensesYear1 INTEGER,
  OnCampusOtherExpensesYear2 INTEGER,
  OnCampusOtherExpensesYear3 INTEGER,
  OffCampusNotWithFamilyRoomBoardYear0 INTEGER,
  OffCampusNotWithFamilyRoomBoardYear1 INTEGER,
  OffCampusNotWithFamilyRoomBoardYear2 INTEGER,
  OffCampusNotWithFamilyRoomBoardYear3 INTEGER,
  OffCampusNotWithFamilyOtherExpensesYear0 INTEGER,
  OffCampusNotWithFamilyOtherExpensesYear1 INTEGER,
  OffCampusNotWithFamilyOtherExpensesYear2 INTEGER,
  OffCampusNotWithFamilyOtherExpensesYear3 INTEGER,
  OffCampusWithFamilyOtherExpensesYear0 INTEGER,
  OffCampusWithFamilyOtherExpensesYear1 INTEGER,
  OffCampusWithFamilyOtherExpensesYear2 INTEGER,
  OffCampusWithFamilyOtherExpensesYear3 INTEGER,
  AcademicYear INTEGER,
  IngestedFileName STRING,
  RowNumber INTEGER
);
```

The attributes of the operational data table is categorized into the 3 groups shown below along with a core group.

10.4 Charges by academic branch

These cover different professional colleges

Chiropractic

- Chiropractic_InStateTuition
- Chiropractic_InStateRequiredFees
- Chiropractic_OutOfStateTuition
- Chiropractic_OutOfStateRequiredFees

Medicine

- Medicine_InStateTuition
- Medicine_InStateRequiredFees
- Medicine_OutOfStateTuition
- Medicine_OutOfStateRequiredFees

10.5 Charges by Residency Type

InDistrictTuition

- PublishedInDistrictTuitionYear0
- PublishedInDistrictTuitionYear1
- PublishedInDistrictTuitionYear2
- PublishedInDistrictTuitionFeesYear3

InDistrictFees

- PublishedInDistrictFeesYear0
- PublishedInDistrictFeesYear1
- PublishedInDistrictFeesYear2
- PublishedInDistrictFeesYear3

InDistrictTuitionFees

- PublishedInDistrictTuitionFeesYear0
- PublishedInDistrictTuitionFeesYear1
- PublishedInDistrictTuitionFeesYear2
- PublishedInDistrictTuitionFeesYear3

InStateTuition

- PublishedInStateTuitionYear0
- PublishedInStateTuitionYear1
- PublishedInStateTuitionYear2
- PublishedInStateTuitionYear3

10.6 Institutional Charge by Category

** BooksSupplies **

- BooksSuppliesYear0
- BooksSuppliesYear1
- BooksSuppliesYear2
- BooksSuppliesYear3

** OnCampusRoomBoard **

- OnCampusRoomBoardYear0
- OnCampusRoomBoardYear1
- OnCampusRoomBoardYear2
- OnCampusRoomBoardYear3

10.7 Stored procedure for OD table load

Stored procedure , called **pr_od_InstitutionalCharge_Load** is added to load the OD table with institutional charge records from the staged file.

This stored procedure takes the YEAR parameter and retrieves the records from the staged file for that year.

Here is the script for creating the stored procedure.

```
CREATE OR REPLACE PROCEDURE
        pr_od_InstitutionalCharge_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_command = `
INSERT INTO od_InstitutionalCharge (
InstitutionIdentifier,
InDistrictAverageTuitionFullTimeUndergraduates,
InDistrictRequiredFeesFullTimeUndergraduates,
InDistrictPerCreditHourChargeForPartTimeUndergraduates,
InStateAverageTuitionFullTimeUndergraduates,
InStateRequiredFeesFullTimeUndergraduates,
InStatePerCreditHourChargeForPartTimeUndergraduates,
OutOfStateAverageTuitionFullTimeUndergraduates,
OutOfStateRequiredFeesFullTimeUndergraduates,
OutOfStatePerCreditHourChargeForPartTimeUndergraduates,
InDistrictAverageTuitionFullTimeGraduates,
InDistrictRequiredFeesFullTimeGraduates,
```

```
InDistrictPerCreditHourChargePartTimeGraduates,
InStateAverageTuitionFullTimeGraduates,
InStateRequiredFeesFullTimeGraduates,
InStatePerCreditHourChargePartTimeGraduates,
OutOfStateAverageTuitionFullTimeGraduates,
OutOfStateRequiredFeesFullTimeGraduates,
OutOfStatePerCreditHourChargePartTimeGraduate,
Chiropractic_InStateTuition,
Chiropractic_InStateRequiredFees,
Chiropractic_OutOfStateTuition,
Chiropractic_OutOfStateRequiredFees,
Dentistry_InStateTuition,
Dentistry_InStateRequiredFees,
Dentistry_OutOfStateTuition,
Dentistry_OutOfStateRequiredFees,
Medicine_InStateTuition,
Medicine_InStateRequiredFees,
Medicine_OutOfStateTuition,
Medicine_OutOfStateRequiredFees,
Optometry_InStateTuition,
Optometry_InStateRequiredFees,
Optometry_OutOfStateTuition,
Optometry_OutOfStateRequiredFees,
OsteopathicMedicine_InStateTuition,
OsteopathicMedicine_InStateRequiredFees,
OsteopathicMedicine_OutOfStateTuition,
OsteopathicMedicine_OutOfStateRequiredFees,
Pharmacy_InStateTuition,
Pharmacy_InStateRequiredFees,
Pharmacy_OutOfStateTuition,
Pharmacy_OutOfStateRequiredFees,
Podiatry_InStateTuition,
Podiatry_InStateRequiredFees,
Podiatry_OutOfStateTuition,
Podiatry_OutOfStateRequiredFees,
VeterinaryMedicine_InStateTuition,
VeterinaryMedicine_InStateRequiredFees,
VeterinaryMedicine_OutOfStateTuition,
VeterinaryMedicine_OutOfStateRequiredFees,
Law_InStateTuition,
Law_InStateRequiredFees,
Law_OutOfStateTuition,
Law_OutOfStateRequiredFees,
PublishedInDistrictTuitionYear0,
PublishedInDistrictFeesYear0,
PublishedInDistrictTuitionFeesYear0,
PublishedInDistrictTuitionYear1,
PublishedInDistrictFeesYear1,
PublishedInDistrictTuitionFeesYear1,
PublishedInDistrictTuitionYear2,
PublishedInDistrictFeesYear2,
PublishedInDistrictTuitionFeesYear2,
PublishedInDistrictTuitionYear3,
PublishedInDistrictFeesYear3,
PublishedInDistrictTuitionFeesYear3,
```

```
PublishedInDistrictTuitionYear3GuaranteedPercentIncrease,
PublishedInDistrictFeesYear3GuaranteedPercentIncrease,
PublishedInStateTuitionYear0,
PublishedInStateFeesYear0,
PublishedInStateTuitionFeesYear0,
PublishedInStateTuitionYear1,
PublishedInStateFeesYear1,
PublishedInStateTuitionFeesYear1,
PublishedInStateTuitionYear2,
PublishedInStateFeesYear2,
PublishedInStateTuitionFeesYear2,
PublishedInStateTuitionYear3,
PublishedInStateFeesYear3,
PublishedInStateTuitionFeesYear3,
PublishedInStateTuitionGuaranteedPercentIncrease,
PublishedInStateFeesGuaranteedPercentIncrease,
PublishedOutOfStateTuitionYear0,
PublishedOutOfStateFeesYear0,
PublishedOutOfStateTuitionFeesYear0,
PublishedOutOfStateTuitionYear1,
PublishedOutOfStateFeesYear1,
PublishedOutOfStateTuitionFeesYear1,
PublishedOutOfStateTuitionYear2,
PublishedOutOfStateFeesYear2,
PublishedOutOfStateTuitionFeesYear2,
PublishedOutOfStateTuitionYear3,
PublishedOutOfStateFeesYear3,
PublishedOutOfStateTuitionFeesYear3,
PublishedOutOfStateTuitionGuaranteedPercentIncrease,
PublishedOutOfStateFeesGuaranteedPercentIncrease,
BooksSuppliesYear0,
BooksSuppliesYear1,
BooksSuppliesYear2,
BooksSuppliesYear3,
OnCampusRoomBoardYear0,
OnCampusRoomBoardYear1,
OnCampusRoomBoardYear2,
OnCampusRoomBoardYear3,
OnCampusOtherExpensesYear0,
OnCampusOtherExpensesYear1,
OnCampusOtherExpensesYear2,
OnCampusOtherExpensesYear3,
OffCampusNotWithFamilyRoomBoardYear0,
OffCampusNotWithFamilyRoomBoardYear1,
OffCampusNotWithFamilyRoomBoardYear2,
OffCampusNotWithFamilyRoomBoardYear3,
OffCampusNotWithFamilyOtherExpensesYear0,
OffCampusNotWithFamilyOtherExpensesYear1,
OffCampusNotWithFamilyOtherExpensesYear2,
OffCampusNotWithFamilyOtherExpensesYear3,
OffCampusWithFamilyOtherExpensesYear0,
OffCampusWithFamilyOtherExpensesYear1,
OffCampusWithFamilyOtherExpensesYear2,
OffCampusWithFamilyOtherExpensesYear3,
AcademicYear,
```

```
IngestedFileName,
RowNumber
)
SELECT
$1:UNITID   :: VARCHAR
        AS InstitutionIdentifier,
$1:TUITION1 :: INTEGER
        AS InDistrictAverageTuitionFullTimeUndergraduates,
$1:FEE1     :: INTEGER
        AS InDistrictRequiredFeesFullTimeUndergraduates,
$1:HRCHG1   :: INTEGER
        AS InDistrictPerCreditHourChargeForPartTimeUndergraduates,
$1:TUITION2 :: INTEGER
        AS InStateAverageTuitionFullTimeUndergraduates,
$1:FEE2     :: INTEGER
        AS InStateRequiredFeesFullTimeUndergraduates,
$1:HRCHG2   :: INTEGER
        AS InStatePerCreditHourChargeForPartTimeUndergraduates,
$1:TUITION3 :: INTEGER
        AS OutOfStateAverageTuitionFullTimeUndergraduates,
$1:FEE3     :: INTEGER
        AS OutOfStateRequiredFeesFullTimeUndergraduates,
$1:HRCHG3   :: INTEGER
        AS OutOfStatePerCreditHourChargeForPartTimeUndergraduates,
$1:TUITION5 :: INTEGER
        AS InDistrictAverageTuitionFullTimeGraduates,
$1:FEE5     :: INTEGER
        AS InDistrictRequiredFeesFullTimeGraduates,
$1:HRCHG5   :: INTEGER
    AS InDistrictPerCreditHourChargePartTimeGraduates,
$1:TUITION6 :: INTEGER
    AS InStateAverageTuitionFullTimeGraduates,
$1:FEE6     :: INTEGER
        AS InStateRequiredFeesFullTimeGraduates,
$1:HRCHG6   :: INTEGER
        AS InStatePerCreditHourChargePartTimeGraduates,
$1:TUITION7 :: INTEGER
        AS OutOfStateAverageTuitionFullTimeGraduates,
$1:FEE7     :: INTEGER
        AS OutOfStateRequiredFeesFullTimeGraduates,
$1:HRCHG7   :: INTEGER
        AS OutOfStatePerCreditHourChargePartTimeGraduate,
$1:ISPROF1  :: INTEGER
        AS Chiropractic_InStateTuition,
$1:ISPFEE1  :: INTEGER
        AS Chiropractic_InStateRequiredFees,
$1:OSPROF1  :: INTEGER
        AS Chiropractic_OutOfStateTuition,
$1:OSPFEE1  :: INTEGER
        AS Chiropractic_OutOfStateRequiredFees,
$1:ISPROF2  :: INTEGER
        AS Dentistry_InStateTuition,
$1:ISPFEE2  :: INTEGER
        AS Dentistry_InStateRequiredFees,
$1:OSPROF2  :: INTEGER
```

```
        AS Dentistry_OutOfStateTuition,
$1:OSPFEE2  :: INTEGER
        AS Dentistry_OutOfStateRequiredFees,
$1:ISPROF3  :: INTEGER
        AS Medicine_InStateTuition,
$1:ISPFEE3  :: INTEGER
        AS Medicine_InStateRequiredFees,
$1:OSPROF3  :: INTEGER
        AS Medicine_OutOfStateTuition,
$1:OSPFEE3  :: INTEGER
        AS Medicine_OutOfStateRequiredFees,
$1:ISPROF4  :: INTEGER
        AS Optometry_InStateTuition,
$1:ISPFEE4  :: INTEGER
        AS Optometry_InStateRequiredFees,
$1:OSPROF4  :: INTEGER
        AS Optometry_OutOfStateTuition,
$1:OSPFEE4  :: INTEGER
        AS Optometry_OutOfStateRequiredFees,
$1:ISPROF5  :: INTEGER
        AS OsteopathicMedicine_InStateTuition,
$1:ISPFEE5  :: INTEGER
        AS OsteopathicMedicine_InStateRequiredFees,
$1:OSPROF5  :: INTEGER
        AS OsteopathicMedicine_OutOfStateTuition,
$1:OSPFEE5  :: INTEGER
        AS OsteopathicMedicine_OutOfStateRequiredFees,
$1:ISPROF6  :: INTEGER
        AS Pharmacy_InStateTuition,
$1:ISPFEE6  :: INTEGER
        AS Pharmacy_InStateRequiredFees,
$1:OSPROF6  :: INTEGER
        AS Pharmacy_OutOfStateTuition,
$1:OSPFEE6  :: INTEGER
        AS Pharmacy_OutOfStateRequiredFees,
$1:ISPROF7  :: INTEGER
        AS Podiatry_InStateTuition,
$1:ISPFEE7  :: INTEGER
        AS Podiatry_InStateRequiredFees,
$1:OSPROF7  :: INTEGER
        AS Podiatry_OutOfStateTuition,
$1:OSPFEE7  :: INTEGER
        AS Podiatry_OutOfStateRequiredFees,
$1:ISPROF8  :: INTEGER
        AS VeterinaryMedicine_InStateTuition,
$1:ISPFEE8  :: INTEGER
        AS VeterinaryMedicine_InStateRequiredFees,
$1:OSPROF8  :: INTEGER
        AS VeterinaryMedicine_OutOfStateTuition,
$1:OSPFEE8  :: INTEGER
        AS VeterinaryMedicine_OutOfStateRequiredFees,
$1:ISPROF9  :: INTEGER
        AS Law_InStateTuition,
$1:ISPFEE9  :: INTEGER
        AS Law_InStateRequiredFees,
```

```
$1:OSPROF9  :: INTEGER
        AS Law_OutOfStateTuition,
$1:OSPFEE9  :: INTEGER
        AS Law_OutOfStateRequiredFees,
$1:CHG1AT0  :: INTEGER
        AS PublishedInDistrictTuitionYear0,
$1:CHG1AF0  :: INTEGER
        AS PublishedInDistrictFeesYear0,
$1:CHG1AY0  :: INTEGER
        AS PublishedInDistrictTuitionFeesYear0,
$1:CHG1AT1  :: INTEGER
        AS PublishedInDistrictTuitionYear1,
$1:CHG1AF1  :: INTEGER
        AS PublishedInDistrictFeesYear1,
$1:CHG1AY1  :: INTEGER
        AS PublishedInDistrictTuitionFeesYear1,
$1:CHG1AT2  :: INTEGER
        AS PublishedInDistrictTuitionYear2,
$1:CHG1AF2  :: INTEGER
        AS PublishedInDistrictFeesYear2,
$1:CHG1AY2  :: INTEGER
        AS PublishedInDistrictTuitionFeesYear2,
$1:CHG1AT3  :: INTEGER
        AS PublishedInDistrictTuitionYear3,
$1:CHG1AF3  :: INTEGER
        AS PublishedInDistrictFeesYear3,
$1:CHG1AY3  :: INTEGER
        AS PublishedInDistrictTuitionFeesYear3,
$1:CHG1TGTD :: INTEGER
        AS PublishedInDistrictTuitionYear3GuaranteedPercentIncrease,
$1:CHG1FGTD :: INTEGER
        AS PublishedInDistrictFeesYear3GuaranteedPercentIncrease,
$1:CHG2AT0  :: INTEGER
        AS PublishedInStateTuitionYear0,
$1:CHG2AF0  :: INTEGER
        AS PublishedInStateFeesYear0,
$1:CHG2AY0  :: INTEGER
        AS PublishedInStateTuitionFeesYear0,
$1:CHG2AT1  :: INTEGER
        AS PublishedInStateTuitionYear1,
$1:CHG2AF1  :: INTEGER
        AS PublishedInStateFeesYear1,
$1:CHG2AY1  :: INTEGER
        AS PublishedInStateTuitionFeesYear1,
$1:CHG2AT2  :: INTEGER
        AS PublishedInStateTuitionYear2,
$1:CHG2AF2  :: INTEGER
        AS PublishedInStateFeesYear2,
$1:CHG2AY2  :: INTEGER
        AS PublishedInStateTuitionFeesYear2,
$1:CHG2AT3  :: INTEGER
        AS PublishedInStateTuitionYear3,
$1:CHG2AF3  :: INTEGER
        AS PublishedInStateFeesYear3,
$1:CHG2AY3  :: INTEGER
```

```
        AS PublishedInStateTuitionFeesYear3,
$1:CHG2TGTD :: INTEGER
        AS PublishedInStateTuitionGuaranteedPercentIncrease,
$1:CHG2FGTD :: INTEGER
        AS PublishedInStateFeesGuaranteedPercentIncrease,
$1:CHG3AT0  :: INTEGER
        AS PublishedOutOfStateTuitionYear0,
$1:CHG3AF0  :: INTEGER
        AS PublishedOutOfStateFeesYear0,
$1:CHG3AY0  :: INTEGER
        AS PublishedOutOfStateTuitionFeesYear0,
$1:CHG3AT1  :: INTEGER
        AS PublishedOutOfStateTuitionYear1,
$1:CHG3AF1  :: INTEGER
        AS PublishedOutOfStateFeesYear1,
$1:CHG3AY1  :: INTEGER
        AS PublishedOutOfStateTuitionFeesYear1,
$1:CHG3AT2  :: INTEGER
        AS PublishedOutOfStateTuitionYear2,
$1:CHG3AF2  :: INTEGER
        AS PublishedOutOfStateFeesYear2,
$1:CHG3AY2  :: INTEGER
        AS PublishedOutOfStateTuitionFeesYear2,
$1:CHG3AT3  :: INTEGER
        AS PublishedOutOfStateTuitionYear3,
$1:CHG3AF3  :: INTEGER
        AS PublishedOutOfStateFeesYear3,
$1:CHG3AY3  :: INTEGER
        AS PublishedOutOfStateTuitionFeesYear3,
$1:CHG3TGTD :: INTEGER
        AS PublishedOutOfStateTuitionGuaranteedPercentIncrease,
$1:CHG3FGTD :: INTEGER
        AS PublishedOutOfStateFeesGuaranteedPercentIncrease,
$1:CHG4AY0  :: INTEGER
        AS BooksSuppliesYear0,
$1:CHG4AY1  :: INTEGER
        AS BooksSuppliesYear1,
$1:CHG4AY2  :: INTEGER
        AS BooksSuppliesYear2,
$1:CHG4AY3  :: INTEGER
        AS BooksSuppliesYear3,
$1:CHG5AY0  :: INTEGER
        AS OnCampusRoomBoardYear0,
$1:CHG5AY1  :: INTEGER
        AS OnCampusRoomBoardYear1,
$1:CHG5AY2  :: INTEGER
        AS OnCampusRoomBoardYear2,
$1:CHG5AY3  :: INTEGER
        AS OnCampusRoomBoardYear3,
$1:CHG6AY0  :: INTEGER
        AS OnCampusOtherExpensesYear0,
$1:CHG6AY1  :: INTEGER
        AS OnCampusOtherExpensesYear1,
$1:CHG6AY2  :: INTEGER
        AS OnCampusOtherExpensesYear2,
```

```
$1:CHG6AY3  :: INTEGER
        AS OnCampusOtherExpensesYear3,
$1:CHG7AY0  :: INTEGER
        AS OffCampusNotWithFamilyRoomBoardYear0,
$1:CHG7AY1  :: INTEGER
        AS OffCampusNotWithFamilyRoomBoardYear1,
$1:CHG7AY2  :: INTEGER
        AS OffCampusNotWithFamilyRoomBoardYear2,
$1:CHG7AY3  :: INTEGER
        AS OffCampusNotWithFamilyRoomBoardYear3,
$1:CHG8AY0  :: INTEGER
        AS OffCampusNotWithFamilyOtherExpensesYear0,
$1:CHG8AY1  :: INTEGER
        AS OffCampusNotWithFamilyOtherExpensesYear1,
$1:CHG8AY2  :: INTEGER
        AS OffCampusNotWithFamilyOtherExpensesYear2,
$1:CHG8AY3  :: INTEGER
        AS OffCampusNotWithFamilyOtherExpensesYear3,
$1:CHG9AY0  :: INTEGER
        AS OffCampusWithFamilyOtherExpensesYear0,
$1:CHG9AY1  :: INTEGER
        AS OffCampusWithFamilyOtherExpensesYear1,
$1:CHG9AY2  :: INTEGER
        AS OffCampusWithFamilyOtherExpensesYear2,
$1:CHG9AY3  :: INTEGER
        AS OffCampusWithFamilyOtherExpensesYear3,
LEFT(MetaData$FileName, 4) AS AcademicYear,
metadata$filename IngestedFileName,
metadata$file_row_number RowNumber
FROM
@IPEDS_IC
 WHERE LEFT(MetaData$FileName,4) = ` + YEAR.toString()+`;`

    try {
        snowflake.execute (
          {sqlText: sql_command}
          );
        }
    catch (err) {
        return "Failed: " + err;
        }
    return "success.";

$$
;
```

The procedure is called for processing various years as below.

```
TRUNCATE TABLE od_InstitutionalCharge;

CALL pr_od_InstitutionalCharge_Load(2017::FLOAT);
CALL pr_od_InstitutionalCharge_Load(2018::FLOAT);
CALL pr_od_InstitutionalCharge_Load(2019::FLOAT);
```

We have added following the entities to the servicing layer. These entities will be populated in the transformation layer.

- InstitutionalCharge
- InstitutionalChargeByAcademicBranch
- InstitutionalChargeByResidencyType
- InstitutionalChargeByCategory

10.8 Servicing layer data structure

While processing institutional charge dataset, datatables are normalized from OD layer and mapped to the servicing layer.

The following script are be used to create these tables.

InstitutionalChargeEntity:

```
CREATE OR REPLACE TABLE InstitutionalCharge (
  AcademicInstitutionUniqueDWSID INTEGER NOT NULL ,
  AcademicYear INTEGER NOT NULL,
  InstitutionIdentifier INTEGER,
  InDistrictAverageTuitionFullTimeUndergraduates INTEGER,
  InDistrictRequiredFeesFullTimeUndergraduates INTEGER,
  InDistrictPerCreditHourChargeForPartTimeUndergraduates INTEGER,
  InStateAverageTuitionFullTimeUndergraduates INTEGER,
  InStateRequiredFeesFullTimeUndergraduates INTEGER,
  InStatePerCreditHourChargeForPartTimeUndergraduates INTEGER,
  OutOfStateAverageTuitionFullTimeUndergraduates INTEGER,
  OutOfStateRequiredFeesFullTimeUndergraduates INTEGER,
  OutOfStatePerCreditHourChargeForPartTimeUndergraduates INTEGER,
  InDistrictAverageTuitionFullTimeGraduates INTEGER,
  InDistrictRequiredFeesFullTimeGraduates INTEGER,
  InDistrictPerCreditHourChargePartTimeGraduates INTEGER,
  InStateAverageTuitionFullTimeGraduates INTEGER,
  InStateRequiredFeesFullTimeGraduates INTEGER,
  InStatePerCreditHourChargePartTimeGraduates INTEGER,
  OutOfStateAverageTuitionFullTimeGraduates INTEGER,
  OutOfStateRequiredFeesFullTimeGraduates INTEGER,
  OutOfStatePerCreditHourChargePartTimeGraduate INTEGER,
  PublishedInStateTuitionGuaranteedPercentIncrease INTEGER,
  PublishedInStateFeesGuaranteedPercentIncrease INTEGER,
  PublishedOutOfStateTuitionGuaranteedPercentIncrease INTEGER,
  PublishedOutOfStateFeesGuaranteedPercentIncrease INTEGER
  );
```

InstitutionalChargeByAcademicBranchEntity:

```
 CREATE OR REPLACE TABLE InstitutionalChargeByAcademicBranch (
  AcademicInstitutionUniqueDWSID INTEGER NOT NULL ,
  InstitutionIdentifier INTEGER,
  AcademicYear  INTEGER NOT NULL,
  BranchName varchar(100) NOT null,
  ChargeType Varchar(100) NOT null,
  Amount  INTEGER );
```

InstitutionalChargeByResidencyTypeEntity:

```
CREATE OR REPLACE TABLE InstitutionalChargeByResidencyType (
  AcademicInstitutionUniqueDWSID INTEGER NOT NULL,
  InstitutionIdentifier INTEGER,
  AcademicYear INTEGER NOT NULL,
  ResidencyType varchar(100) NOT NULL,
  PublishedYear INTEGER NOT NULL,
  Amount  INTEGER );
```

InstitutionalChargeByCategoryEntity:

```
CREATE OR REPLACE TABLE InstitutionalChargeByCategory (
  AcademicInstitutionUniqueDWSID INTEGER NOT NULL,
  InstitutionIdentifier INTEGER,
  AcademicYear INTEGER NOT NULL,
  CategoryType varchar(100) NOT NULL,
  PublishedYear INTEGER NOT NULL,
  Amount  INTEGER );
```

10.9 Unpivot data to load into the Servicing layer

In our case, data is stored in the OD layer in a pivoted record structure. We will need to unpivot these attributes to load the servicing layer.

We can unpivot data structure in a traditional SQL environment by creating a union on underlying attributes.

Another flexible approach will be to store JSON object in the OD layer and simplify the load. As new categories (attributes) are added, it is easy to manage. Data structures are easy to digest for programmers. We will elaborate on this in the next section.

The following views are created to transform columns to rows. These views will be used in load stored procedures.

10.9.1 Institutional Charge by Academic Branch

v_od_InstitutionalChargeByAcademicBranch is created to convert the columns from underlying OD dataset to rows for data processing layer. The view outputs one record per academic branch like chiropractic, medicine, OsteopathicMedicine, ... Attribute columns from source are categorized into tution and fees.

```
CREATE
        OR REPLACE VIEW v_od_InstitutionalChargeByAcademicBranch AS

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Chiropractic' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.Chiropractic_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
```

```
        ,'Chiropractic'
        ,'InStateRequiredFees'
        ,S.Chiropractic_InStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Chiropractic'
        ,'OutOfStateTuition'
        ,S.Chiropractic_OutOfStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Chiropractic'
        ,'OutOfStateRequiredFees'
        ,S.Chiropractic_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--Dentistry
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Dentistry' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.Dentistry_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Dentistry'
        ,'InStateRequiredFees'
        ,S.Dentistry_InStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Dentistry'
        ,'OutOfStateTuition'
        ,S.Dentistry_OutOfStateTuition
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
```

```
        ,'Dentistry'
        ,'OutOfStateRequiredFees'
        ,S.Dentistry_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--Medicine
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Medicine' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.Medicine_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Medicine'
        ,'InStateRequiredFees'
        ,S.Medicine_InStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Medicine'
        ,'OutOfStateTuition'
        ,S.Medicine_OutOfStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Medicine'
        ,'OutOfStateRequiredFees'
        ,S.Medicine_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--Optometry
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Optometry' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.Optometry_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
```

```
        ,S.AcademicYear
        ,'Optometry'
        ,'InStateRequiredFees'
        ,S.Optometry_InStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Optometry'
        ,'OutOfStateTuition'
        ,S.Optometry_OutOfStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Optometry'
        ,'OutOfStateRequiredFees'
        ,S.Optometry_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--OsteopathicMedicine
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OsteopathicMedicine' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.OsteopathicMedicine_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OsteopathicMedicine'
        ,'InStateRequiredFees'
        ,S.OsteopathicMedicine_InStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OsteopathicMedicine'
        ,'OutOfStateTuition'
        ,S.OsteopathicMedicine_OutOfStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
```

```
        ,S.AcademicYear
        ,'OsteopathicMedicine'
        ,'OutOfStateRequiredFees'
        ,S.OsteopathicMedicine_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--Pharmacy
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Pharmacy' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.Pharmacy_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Pharmacy'
        ,'InStateRequiredFees'
        ,S.Pharmacy_InStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Pharmacy'
        ,'OutOfStateTuition'
        ,S.Pharmacy_OutOfStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Pharmacy'
        ,'OutOfStateRequiredFees'
        ,S.Pharmacy_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--Podiatry
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Podiatry' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.Podiatry_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL
```

```sql
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Podiatry'
        ,'InStateRequiredFees'
        ,S.Podiatry_InStateRequiredFees
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Podiatry'
        ,'OutOfStateTuition'
        ,S.Podiatry_OutOfStateTuition
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Podiatry'
        ,'OutOfStateRequiredFees'
        ,S.Podiatry_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--VeterinaryMedicine
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'VeterinaryMedicine' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.VeterinaryMedicine_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'VeterinaryMedicine'
        ,'InStateRequiredFees'
        ,S.VeterinaryMedicine_InStateRequiredFees
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'VeterinaryMedicine'
        ,'OutOfStateTuition'
        ,S.VeterinaryMedicine_OutOfStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL
```

```
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'VeterinaryMedicine'
        ,'OutOfStateRequiredFees'
        ,S.VeterinaryMedicine_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--Law
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Law' AS BranchName
        ,'InStateTuition' AS ChargeType
        ,S.Law_InStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Law'
        ,'InStateRequiredFees'
        ,S.Law_InStateRequiredFees AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Law'
        ,'OutOfStateTuition'
        ,S.Law_OutOfStateTuition AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'Law'
        ,'OutOfStateRequiredFees'
        ,S.Law_OutOfStateRequiredFees AS Amount
FROM od_InstitutionalCharge S;
```

10.9.2 Institutional Charge by Residency Type

v_od_InstitutionalChargeByResidencyType is similar to the v_od_InstitutionalChargeByAcademicBranch. It is also converting columns to rows for data processing layer. This view output one record per: in district, in state, out of state residency type. The view provides tution information for complete course work like year 0 tution, year 1 tution.

```
CREATE
        OR REPLACE VIEW v_od_InstitutionalChargeByResidencyType AS
```

```
--InDistrict
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuition' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedInDistrictTuitionYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictFees' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedInDistrictFeesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuitionFees' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedInDistrictTuitionFeesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuition' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedInDistrictTuitionYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictFees' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedInDistrictFeesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuitionFees' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedInDistrictTuitionFeesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL
```

```
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuition' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedInDistrictTuitionYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictFees' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedInDistrictFeesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuitionFees' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedInDistrictTuitionFeesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuition' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedInDistrictTuitionYear3 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictFees' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedInDistrictFeesYear3 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'DistrictTuitionFees' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedInDistrictTuitionFeesYear3 AS Amount
FROM od_InstitutionalCharge S
--InState

UNION ALL
```

```sql
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateTuition' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedInStateTuitionYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateFees' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedInStateFeesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateTuitionFees' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedInStateTuitionFeesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateTuition' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedInStateTuitionYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateFees' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedInStateFeesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateTuitionFees' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedInStateTuitionFeesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
```

```
        ,S.AcademicYear
        ,'InStateTuition' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedInStateTuitionYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateFees' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedInStateFeesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateTuitionFees' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedInStateTuitionFeesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateTuition' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedInStateTuitionYear3 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateFees' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedInStateFeesYear3 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'InStateTuitionFees' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedInStateTuitionFeesYear3 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

--OutOfState
SELECT S.InstitutionIdentifier
```

```
        ,S.AcademicYear
        ,'OutOfStateTuition' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedOutOfStateTuitionYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateFees' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedOutOfStateFeesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateTuitionFees' AS ResidencyType
        ,0 AS PublishedYear
        ,S.PublishedOutOfStateTuitionFeesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateTuition' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedOutOfStateTuitionYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateFees' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedOutOfStateFeesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateTuitionFees' AS ResidencyType
        ,1 AS PublishedYear
        ,S.PublishedOutOfStateTuitionFeesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
```

```
        ,'OutOfStateTuition' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedOutOfStateTuitionYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateFees' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedOutOfStateFeesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateTuitionFees' AS ResidencyType
        ,2 AS PublishedYear
        ,S.PublishedOutOfStateTuitionFeesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateTuition' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedOutOfStateTuitionYear3 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateFees' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedOutOfStateFeesYear3 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OutOfStateTuitionFees' AS ResidencyType
        ,3 AS PublishedYear
        ,S.PublishedOutOfStateTuitionFeesYear3 AS Amount
FROM od_InstitutionalCharge S;
```

10.9.3 Institutional Charge by Category

v_od_InstitutionalChargeByCategory is built with the same concepts as v_od_InstitutionalChargeByResidencyType and v_od_InstitutionalChargeByAcademicBranch. This view provides more granular records to detail out expenses for books supplies, rent, food, and other types of expenses during college education.

```
CREATE
        OR REPLACE VIEW v_od_InstitutionalChargeByCategory AS

--BooksSupplies
SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'BooksSupplies' AS CategoryType
        ,0 AS PublishedYear
        ,S.BooksSuppliesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'BooksSupplies' AS CategoryType
        ,1 AS PublishedYear
        ,S.BooksSuppliesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'BooksSupplies' AS CategoryType
        ,2 AS PublishedYear
        ,S.BooksSuppliesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'BooksSupplies' AS CategoryType
        ,3 AS PublishedYear
        ,S.BooksSuppliesYear3 AS Amount
FROM od_InstitutionalCharge S
--OnCampusRoomBoard

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusRoomBoard' AS CategoryType
        ,0 AS PublishedYear
        ,S.OnCampusRoomBoardYear0 AS Amount
FROM od_InstitutionalCharge S
```

```
UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusRoomBoard' AS CategoryType
        ,1 AS PublishedYear
        ,S.OnCampusRoomBoardYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusRoomBoard' AS CategoryType
        ,2 AS PublishedYear
        ,S.OnCampusRoomBoardYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusRoomBoard' AS CategoryType
        ,3 AS PublishedYear
        ,S.OnCampusRoomBoardYear3 AS Amount
FROM od_InstitutionalCharge S
--OnCampusOtherExpenses

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusOtherExpenses' AS CategoryType
        ,0 AS PublishedYear
        ,S.OnCampusOtherExpensesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusOtherExpenses' AS CategoryType
        ,1 AS PublishedYear
        ,S.OnCampusOtherExpensesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusOtherExpenses' AS CategoryType
        ,2 AS PublishedYear
        ,S.OnCampusOtherExpensesYear2 AS Amount
FROM od_InstitutionalCharge S
```

```
UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OnCampusOtherExpenses' AS CategoryType
        ,3 AS PublishedYear
        ,S.OnCampusOtherExpensesYear3 AS Amount
FROM od_InstitutionalCharge S
--OffCampusNotWithFamilyRoomBoard

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyRoomBoard' AS CategoryType
        ,0 AS PublishedYear
        ,S.OffCampusNotWithFamilyRoomBoardYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyRoomBoard' AS CategoryType
        ,1 AS PublishedYear
        ,S.OffCampusNotWithFamilyRoomBoardYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyRoomBoard' AS CategoryType
        ,2 AS PublishedYear
        ,S.OffCampusNotWithFamilyRoomBoardYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyRoomBoard' AS CategoryType
        ,3 AS PublishedYear
        ,S.OffCampusNotWithFamilyRoomBoardYear3 AS Amount
FROM od_InstitutionalCharge S
--OffCampusNotWithFamilyOtherExpenses

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyOtherExpenses' AS CategoryType
        ,0 AS PublishedYear
        ,S.OffCampusNotWithFamilyOtherExpensesYear0 AS Amount
FROM od_InstitutionalCharge S
```

```
UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyOtherExpenses' AS CategoryType
        ,1 AS PublishedYear
        ,S.OffCampusNotWithFamilyOtherExpensesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyOtherExpenses' AS CategoryType
        ,2 AS PublishedYear
        ,S.OffCampusNotWithFamilyOtherExpensesYear2 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusNotWithFamilyOtherExpenses' AS CategoryType
        ,3 AS PublishedYear
        ,S.OffCampusNotWithFamilyOtherExpensesYear3 AS Amount
FROM od_InstitutionalCharge S
--OffCampusWithFamilyOtherExpenses

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusWithFamilyOtherExpenses' AS CategoryType
        ,0 AS PublishedYear
        ,S.OffCampusWithFamilyOtherExpensesYear0 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusWithFamilyOtherExpenses' AS CategoryType
        ,1 AS PublishedYear
        ,S.OffCampusWithFamilyOtherExpensesYear1 AS Amount
FROM od_InstitutionalCharge S

UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusWithFamilyOtherExpenses' AS CategoryType
        ,2 AS PublishedYear
        ,S.OffCampusWithFamilyOtherExpensesYear2 AS Amount
FROM od_InstitutionalCharge S
```

```
UNION ALL

SELECT S.InstitutionIdentifier
        ,S.AcademicYear
        ,'OffCampusWithFamilyOtherExpenses' AS CategoryType
        ,3 AS PublishedYear
        ,S.OffCampusWithFamilyOtherExpensesYear3 AS Amount
FROM od_InstitutionalCharge S;
```

10.10 Data load into the Servicing layer

We will be using the OD table od_InstitutionalCharge and the above 3 views to finally land data to our destination. There are four stored procedures, that use these data sources and land it. All these SP will take academic year parameter. For the given year, they will delete existing data from the destination tables and move data from the OD layer.

The following stored procedures code is used to create these objects.

10.10.1 Institutional Charge (general) load

```
CREATE OR REPLACE PROCEDURE
        pr_InstitutionalCharge_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_DelCommand = `DELETE FROM InstitutionalCharge
        WHERE AcademicYear = `+ YEAR.toString() +`;`
var sql_Command =
`INSERT INTO
InstitutionalCharge
(AcademicInstitutionUniqueDWSID,
InstitutionIdentifier,
AcademicYear,
InDistrictAverageTuitionFullTimeUndergraduates,
InDistrictRequiredFeesFullTimeUndergraduates,
InDistrictPerCreditHourChargeForPartTimeUndergraduates,
InStateAverageTuitionFullTimeUndergraduates,
InStateRequiredFeesFullTimeUndergraduates,
InStatePerCreditHourChargeForPartTimeUndergraduates,
OutOfStateAverageTuitionFullTimeUndergraduates,
OutOfStateRequiredFeesFullTimeUndergraduates,
OutOfStatePerCreditHourChargeForPartTimeUndergraduates,
InDistrictAverageTuitionFullTimeGraduates,
InDistrictRequiredFeesFullTimeGraduates,
InDistrictPerCreditHourChargePartTimeGraduates,
InStateAverageTuitionFullTimeGraduates,
InStateRequiredFeesFullTimeGraduates,
InStatePerCreditHourChargePartTimeGraduates,
OutOfStateAverageTuitionFullTimeGraduates,
OutOfStateRequiredFeesFullTimeGraduates,
OutOfStatePerCreditHourChargePartTimeGraduate,
PublishedInStateTuitionGuaranteedPercentIncrease,
PublishedInStateFeesGuaranteedPercentIncrease,
PublishedOutOfStateTuitionGuaranteedPercentIncrease,
PublishedOutOfStateFeesGuaranteedPercentIncrease
)
SELECT
IFNULL(D.AcademicInstitutionUniqueDWSID, - 1),
S.InstitutionIdentifier,
S.AcademicYear,
S.InDistrictAverageTuitionFullTimeUndergraduates,
S.InDistrictRequiredFeesFullTimeUndergraduates,
S.InDistrictPerCreditHourChargeForPartTimeUndergraduates,
S.InStateAverageTuitionFullTimeUndergraduates,
S.InStateRequiredFeesFullTimeUndergraduates,
S.InStatePerCreditHourChargeForPartTimeUndergraduates,
S.OutOfStateAverageTuitionFullTimeUndergraduates,
S.OutOfStateRequiredFeesFullTimeUndergraduates,
S.OutOfStatePerCreditHourChargeForPartTimeUndergraduates,
S.InDistrictAverageTuitionFullTimeGraduates,
S.InDistrictRequiredFeesFullTimeGraduates,
```

```
S.InDistrictPerCreditHourChargePartTimeGraduates,
S.InStateAverageTuitionFullTimeGraduates,
S.InStateRequiredFeesFullTimeGraduates,
S.InStatePerCreditHourChargePartTimeGraduates,
S.OutOfStateAverageTuitionFullTimeGraduates,
S.OutOfStateRequiredFeesFullTimeGraduates,
S.OutOfStatePerCreditHourChargePartTimeGraduate,
S.PublishedInStateTuitionGuaranteedPercentIncrease,
S.PublishedInStateFeesGuaranteedPercentIncrease,
S.PublishedOutOfStateTuitionGuaranteedPercentIncrease,
S.PublishedOutOfStateFeesGuaranteedPercentIncrease
FROM
  od_InstitutionalCharge S
LEFT OUTER JOIN AcademicInstitution D
        ON S.InstitutionIdentifier = D.InstitutionIdentifier
WHERE S.AcademicYear = ` + YEAR.toString() +`;`

var arrSqlCmd =[2];
arrSqlCmd[0] = sql_DelCommand
arrSqlCmd[1] = sql_Command
var sql_ExecCommand ='';
for (var j=0;j<arrSqlCmd.length; j++)
{
        sql_ExecCommand = arrSqlCmd[j]
        try
        {
                snowflake.execute(
                        {sqlText: sql_ExecCommand}
                );
  }
        catch(err)
        {
                return "Failed :" + err;
        }
}
return "success"
$$;
```

10.10.2 Institutional Charge by academic branch load

```
CREATE OR REPLACE PROCEDURE
        pr_InstitutionalChargeByAcademicBranch_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_DelCommand =
`DELETE FROM InstitutionalChargeByAcademicBranch
        WHERE AcademicYear = `+ YEAR.toString() +`;`
var sql_Command =
`INSERT INTO
   InstitutionalChargeByAcademicBranch (
          AcademicInstitutionUniqueDWSID,
          InstitutionIdentifier,
          AcademicYear,
          BranchName,
          ChargeType,
          Amount
    )
   SELECT
        IFNULL(D.AcademicInstitutionUniqueDWSID,-1),
                S.InstitutionIdentifier,
                S.AcademicYear,
                S.BranchName,
                S.ChargeType,
                S.Amount
   FROM
      V_od_InstitutionalChargeByAcademicBranch S
    LEFT OUTER JOIN AcademicInstitution D
                ON S.InstitutionIdentifier = D.InstitutionIdentifier
    WHERE S.AcademicYear = ` + YEAR.toString() +`;`

    var arrSqlCmd =[2];
    arrSqlCmd[0] = sql_DelCommand
    arrSqlCmd[1] = sql_Command
    var sql_ExecCommand ='';
    for (var j=0;j<arrSqlCmd.length; j++)
    {
        sql_ExecCommand = arrSqlCmd[j]
        try
        {
            snowflake.execute(
                {sqlText: sql_ExecCommand}
            );
      }
        catch(err)
        {
            return "Failed :" + err;
        }
    }
    return "success"
$$;
```

10.10.3 Institutional Charge by residency type load

```
CREATE
        OR REPLACE PROCEDURE
        pr_InstitutionalChargeByResidencyType_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_DelCommand =
        `DELETE FROM InstitutionalChargeByResidencyType
                WHERE AcademicYear = `+ YEAR.toString() +`;`
var sql_Command =
`INSERT INTO
   InstitutionalChargeByResidencyType (
          AcademicInstitutionUniqueDWSID,
          InstitutionIdentifier,
          AcademicYear,
          ResidencyType,
          PublishedYear,
          Amount
    )
   SELECT
        IFNULL(D.AcademicInstitutionUniqueDWSID,-1),
                S.InstitutionIdentifier,
                S.AcademicYear,
                S.ResidencyType,
                S.PublishedYear,
                S.Amount
   FROM
      V_od_InstitutionalChargeByResidencyType S
    LEFT OUTER JOIN AcademicInstitution D
                ON S.InstitutionIdentifier = D.InstitutionIdentifier
    WHERE S.AcademicYear = ` + YEAR.toString() +`;`

    var arrSqlCmd =[2];
    arrSqlCmd[0] = sql_DelCommand
    arrSqlCmd[1] = sql_Command
    var sql_ExecCommand ='';
    for (var j=0;j<arrSqlCmd.length; j++)
    {
        sql_ExecCommand = arrSqlCmd[j]
        try
        {
            snowflake.execute(
                {sqlText: sql_ExecCommand}
            );
      }
        catch(err)
        {
            return "Failed :" + err;
        }
    }
```

```
    return "success"
$$;
```

10.10.4 Institutional Charge By category load

```
CREATE OR REPLACE PROCEDURE
        pr_InstitutionalChargeByCategory_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_DelCommand =
        `DELETE FROM InstitutionalChargeByCategory
                WHERE AcademicYear = `+ YEAR.toString() +`;`
var sql_Command =
`INSERT INTO
   InstitutionalChargeByCategory (
          AcademicInstitutionUniqueDWSID,
          InstitutionIdentifier,
          AcademicYear,
          CategoryType,
          PublishedYear,
          Amount
    )
   SELECT
        IFNULL(D.AcademicInstitutionUniqueDWSID,-1),
                S.InstitutionIdentifier,
                S.AcademicYear,
                S.CategoryType,
                S.PublishedYear,
                S.Amount
   FROM
      V_Od_InstitutionalChargeByCategory S
    LEFT OUTER JOIN AcademicInstitution D
                ON S.InstitutionIdentifier = D.InstitutionIdentifier
    WHERE S.AcademicYear = ` + YEAR.toString() +`;`

    var arrSqlCmd =[2];
    arrSqlCmd[0] = sql_DelCommand
    arrSqlCmd[1] = sql_Command
    var sql_ExecCommand ='';
    for (var j=0;j<arrSqlCmd.length; j++)
    {
        sql_ExecCommand = arrSqlCmd[j]
        try
        {
            snowflake.execute(
                {sqlText: sql_ExecCommand}
            );
      }
        catch(err)
        {
            return "Failed :" + err;
        }
    }
    return "success"
$$;
```

Now data for all these entities associated with institutional charge is loaded with a calling these stored procedures as below:

```
CALL pr_InstitutionalCharge_Load(2017::FLOAT);
CALL pr_InstitutionalCharge_Load(2018::FLOAT);
CALL pr_InstitutionalCharge_Load(2019::FLOAT);

CALL pr_InstitutionalChargeByAcademicBranch_Load(2017::FLOAT);
CALL pr_InstitutionalChargeByAcademicBranch_Load(2018::FLOAT);
CALL pr_InstitutionalChargeByAcademicBranch_Load(2019::FLOAT);


CALL pr_InstitutionalChargeByResidencyType_Load(2017::FLOAT);
CALL pr_InstitutionalChargeByResidencyType_Load(2018::FLOAT);
CALL pr_InstitutionalChargeByResidencyType_Load(2019::FLOAT);



CALL pr_InstitutionalChargeByCategory_Load(2017::FLOAT);
CALL pr_InstitutionalChargeByCategory_Load(2018::FLOAT);
CALL pr_InstitutionalChargeByCategory_Load(2019::FLOAT);
```

10.11 Summary

Using ORC files in Snowflake is as easy as using CSV files. We can perform variety of transformations while moving data from the landing layer to the servicing layer.

> ***Note***
>
> Code available to download at
>
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/source/code/SQL/InstitutionalCharge

11 Parquet Ingestion

11.1 Admission statistics

The admission statistics contain information about the number of applicants by various categories, admission test scores, SAT/ACT percentile scores and so on. These statistics are stored by institution and year. Admission statistics files are received in the Parquet format.

We will learn how to

- Use ingested Parquet files from the staged Snowflake region
- Transform data using the Snowflake JavaScript stored procedures
- Call the stored procedures

The staged file for keeping track of institution admission statistics is **@IPEDS_ADM**. Please note that the columns are selected using **$1 :<column_name>**. This is because data files used are in the Parquet format. We can retrieve the details of a particular year as below.

```
SELECT
$1:UNITID::INTEGER AS
 InstitutionIdentifier,
$1:ADMCON1::INTEGER AS
 SecondarySchoolGPA,
$1:ADMCON2::INTEGER AS
 SecondarySchoolRank,
$1:ADMCON3::INTEGER AS
 SecondarySchoolRecord,
$1:ADMCON4::INTEGER AS
 CompletionOfCollegePreparatoryProgram,
$1:ADMCON5::INTEGER AS
 Recommendations,
$1:ADMCON6::INTEGER AS
 FormalDemonstrationOfCompetencies,
$1:ADMCON7::INTEGER AS
 AdmissionTestScores,
$1:ADMCON8::INTEGER AS
 Toefl,
$1:ADMCON9::INTEGER AS
 OtherTest,
$1:APPLCN::INTEGER AS
 ApplicantsTotal,
$1:APPLCNM::INTEGER AS
 ApplicantsMen,
$1:APPLCNW::INTEGER AS
 ApplicantsWomen,
$1:ADMSSN::INTEGER AS
 AdmissionsTotal,
$1:ADMSSNM::INTEGER AS
 AdmissionsMen,
$1:ADMSSNW::INTEGER AS
 AdmissionsWomen,
$1:ENRLT::INTEGER AS
 EnrolledTotal,
$1:ENRLM::INTEGER AS
```

```
 EnrolledMen,
$1:ENRLW::INTEGER AS
 EnrolledWomen,
$1:ENRLFT::INTEGER AS
 EnrolledFullTimeTotal,
$1:ENRLFTM::INTEGER AS
 EnrolledFullTimeMen,
$1:ENRLFTW::INTEGER AS
 EnrolledFullTimeWomen,
$1:ENRLPT::INTEGER AS
 EnrolledPartTimeTotal,
$1:ENRLPTM::INTEGER AS
 EnrolledPartTimeMen,
$1:ENRLPTW::INTEGER AS
 EnrolledPartTimeWomen,
$1:SATNUM::INTEGER AS
 FirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores,
$1:SATPCT::INTEGER AS
 PercentFirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores,
$1:ACTNUM::INTEGER AS
 FirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores,
$1:ACTPCT::INTEGER AS
 PercentOfFirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores,
$1:SATVR25::INTEGER AS
 SATEvidenceBasedReadingWriting25thPercentileScore,
$1:SATVR75::INTEGER AS
 SATEvidenceBasedReadingWriting75thPercentileScore,
$1:SATMT25::INTEGER AS
 SATMath25thPercentileScore,
$1:SATMT75::INTEGER AS
 SATMath75thPercentileScore,
$1:ACTCM25::INTEGER AS
 ACTComposite25thPercentileScore,
$1:ACTCM75::INTEGER AS
 ACTComposite75thPercentileScore,
$1:ACTEN25::INTEGER AS
 ACTEnglish25thPercentileScore,
$1:ACTEN75::INTEGER AS
 ACTEnglish75thPercentileScore,
$1:ACTMT25::INTEGER AS
 ACTMath25thPercentileScore,
$1:ACTMT75::INTEGER AS
 ACTMath75thPercentileScore,
LEFT(METADATA$FileName,4)::INTEGER AS
 AcademicYear,
metadata$filename::VARCHAR AS
 IngestedFileName,
metadata$file_row_number::INTEGER AS
 RowNumber
FROM
@IPEDS_ADM
WHERE CAST(LEFT(METADATA$FileName,4) AS
 VARCHAR) = '2017';
```

11.2 OD table for Admission Statistics

To keep track of admission statistics by year, we have created an operational data table called **od_AdmissionStat**. We have added 3 additional columns AcademicYear, IngestedFileName, and RowNumber for this OD table as well. When processing the Parquet data, all the partitions from the staged location are used. The staged location metadata also contains part of the partition key. In our case the first 4 characters contains year in a YYYY format.

Snowflake SQL script for creating this table given below.

```
CREATE OR REPLACE TABLE od_AdmissionStat (
InstitutionIdentifier STRING,
SecondarySchoolGPA STRING,
SecondarySchoolRank STRING,
SecondarySchoolRecord STRING,
CompletionOfCollegePreparatoryProgram STRING,
Recommendations STRING,
FormalDemonstrationOfCompetencies STRING,
AdmissionTestScores STRING,
TOEFL STRING,
OtherTest STRING,
ApplicantsTotal STRING,
ApplicantsMen STRING,
ApplicantsWomen STRING,
AdmissionsTotal STRING,
AdmissionsMen STRING,
AdmissionsWomen STRING,
EnrolledTotal STRING,
EnrolledMen STRING,
EnrolledWomen STRING,
EnrolledFullTimeTotal STRING,
EnrolledFullTimeMen STRING,
EnrolledFullTimeWomen STRING,
EnrolledPartTimeTotal STRING,
EnrolledPartTimeMen STRING,
EnrolledPartTimeWomen STRING,
FirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores
        STRING,
PercentFirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores
        STRING,
FirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores
        STRING,
PercentOfFirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores
        STRING,
SATEvidenceBasedReadingWriting25thPercentileScore STRING,
SATEvidenceBasedReadingWriting75thPercentileScore STRING,
SATMath25thPercentileScore STRING,
SATMath75thPercentileScore STRING,
ACTComposite25thPercentileScore STRING,
ACTComposite75thPercentileScore STRING,
ACTEnglish25thPercentileScore STRING,
ACTEnglish75thPercentileScore STRING,
ACTMath25thPercentileScore STRING,
ACTMath75thPercentileScore STRING,
AcademicYear INTEGER,
IngestedFileName STRING,
```

```
RowNumber INTEGER
);
```

11.3 Stored procedure for staging table

Once the staging table is created, we need to populate the table with admission statistics data for each year from the staged file. For that purpose, we've created a stored procedure called **pr_od_AdmissionStat_Load**. This stored procedure takes the YEAR parameter and retrieves the records from the staged file for that year. The parameter YEAR is compared against the first 4 characters of **METADATA$FileName** of the staging file and retrieves records for that year.

Here is the script for creating the stored procedure.

```
CREATE OR REPLACE PROCEDURE pr_od_AdmissionStat_Load(YEAR FLOAT)
RETURNS STRING
LANGUAGE javascript
EXECUTE AS OWNER
AS
$$
var sql_command = `
INSERT INTO od_AdmissionStat
 (
  InstitutionIdentifier,
  SecondarySchoolGPA,
  SecondarySchoolRank,
  SecondarySchoolRecord,
  CompletionOfCollegePreparatoryProgram,
  Recommendations,
  FormalDemonstrationOfCompetencies,
  AdmissionTestScores,
  TOEFL,
  OtherTest,
  ApplicantsTotal,
  ApplicantsMen,
  ApplicantsWomen,
  AdmissionsTotal,
  AdmissionsMen,
  AdmissionsWomen,
  EnrolledTotal,
  EnrolledMen,
  EnrolledWomen,
  EnrolledFullTimeTotal,
  EnrolledFullTimeMen,
  EnrolledFullTimeWomen,
  EnrolledPartTimeTotal,
  EnrolledPartTimeMen,
  EnrolledPartTimeWomen,
  FirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores,
  PercentFirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores,
  FirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores,
  PercentOfFirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores,
  SATEvidenceBasedReadingWriting25thPercentileScore,
  SATEvidenceBasedReadingWriting75thPercentileScore,
  SATMath25thPercentileScore,
```

```
  SATMath75thPercentileScore,
  ACTComposite25thPercentileScore,
  ACTComposite75thPercentileScore,
  ACTEnglish25thPercentileScore,
  ACTEnglish75thPercentileScore,
  ACTMath25thPercentileScore,
  ACTMath75thPercentileScore,
  AcademicYear,
  IngestedFileName,
  RowNumber
)
SELECT
$1:UNITID::INTEGER AS InstitutionIdentifier,
$1:ADMCON1::INTEGER AS SecondarySchoolGPA,
$1:ADMCON2::INTEGER AS SecondarySchoolRank,
$1:ADMCON3::INTEGER AS SecondarySchoolRecord,
$1:ADMCON4::INTEGER AS CompletionOfCollegePreparatoryProgram,
$1:ADMCON5::INTEGER AS Recommendations,
$1:ADMCON6::INTEGER AS FormalDemonstrationOfCompetencies,
$1:ADMCON7::INTEGER AS AdmissionTestScores,
$1:ADMCON8::INTEGER AS Toefl,
$1:ADMCON9::INTEGER AS OtherTest,
$1:APPLCN::INTEGER AS ApplicantsTotal,
$1:APPLCNM::INTEGER AS ApplicantsMen,
$1:APPLCNW::INTEGER AS ApplicantsWomen,
$1:ADMSSN::INTEGER AS AdmissionsTotal,
$1:ADMSSNM::INTEGER AS AdmissionsMen,
$1:ADMSSNW::INTEGER AS AdmissionsWomen,
$1:ENRLT::INTEGER AS EnrolledTotal,
$1:ENRLM::INTEGER AS EnrolledMen,
$1:ENRLW::INTEGER AS EnrolledWomen,
$1:ENRLFT::INTEGER AS EnrolledFullTimeTotal,
$1:ENRLFTM::INTEGER AS EnrolledFullTimeMen,
$1:ENRLFTW::INTEGER AS EnrolledFullTimeWomen,
$1:ENRLPT::INTEGER AS EnrolledPartTimeTotal,
$1:ENRLPTM::INTEGER AS EnrolledPartTimeMen,
$1:ENRLPTW::INTEGER AS EnrolledPartTimeWomen,
$1:SATNUM::INTEGER AS
FirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores,
$1:SATPCT::INTEGER AS
PercentFirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores,
$1:ACTNUM::INTEGER AS
FirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores,
$1:ACTPCT::INTEGER AS
PercentOfFirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores,
$1:SATVR25::INTEGER AS
SATEvidenceBasedReadingWriting25thPercentileScore,
$1:SATVR75::INTEGER AS SATEvidenceBasedReadingWriting75thPercentileScore,
$1:SATMT25::INTEGER AS SATMath25thPercentileScore,
$1:SATMT75::INTEGER AS SATMath75thPercentileScore,
$1:ACTCM25::INTEGER AS ACTComposite25thPercentileScore,
$1:ACTCM75::INTEGER AS ACTComposite75thPercentileScore,
$1:ACTEN25::INTEGER AS ACTEnglish25thPercentileScore,
$1:ACTEN75::INTEGER AS ACTEnglish75thPercentileScore,
$1:ACTMT25::INTEGER AS ACTMath25thPercentileScore,
```

```
$1:ACTMT75::INTEGER AS ACTMath75thPercentileScore,
LEFT(METADATA$FileName,4)::INTEGER AS AcademicYear,
metadata$filename::VARCHAR AS IngestedFileName,
metadata$file_row_number::INTEGER AS RowNumber
FROM
  @IPEDS_ADM
WHERE CAST(LEFT(METADATA$FileName,4) AS VARCHAR) = ` + YEAR.toString()+`;`

    try {
        snowflake.execute (
          {sqlText: sql_command}
          );
        }
    catch (err) {
        return "Failed: " + err;  // Return a success/error indicator.
        }
    return "Success.";  // Return a success/error indicator.
$$
;
```

The OD table is truncated and loaded by calling the stored procedure as below.

```
TRUNCATE TABLE od_AdmissionStat;

CALL pr_od_AdmissionStat_Load(2017::FLOAT);
CALL pr_od_AdmissionStat_Load(2018::FLOAT);
CALL pr_od_AdmissionStat_Load(2019::FLOAT);
```

In a production environment, we would process one file at a time all the way through the servicing layer.

11.4 Admission Statistics Servicing layer

Servicing layer tables or fact tables are mostly final layer tables in a data warehouse environment. These tables are directly consumed by the end users or visualization team. We have created a fact table for admission statistics called **AdmissionStat**. This table has links to the dimension table **AcademicInstitution** using a star schema in dimensional modelling architecture. This is made possible by adding the surrogate key column (**AcademicInstitutionUniqueDWSID**) of the dimension table AcademicInstitution to the fact table AdmissionStat.

Here is the script for creating the table AdmissionStat in Snowflake.

```
DROP TABLE IF EXISTS AdmissionStat;

CREATE TABLE AdmissionStat(
AcademicInstitutionUniqueDWSID INTEGER,
InstitutionIdentifier STRING,
AcademicYear INTEGER,
SecondarySchoolGPA STRING,
SecondarySchoolRank STRING,
SecondarySchoolRecord STRING,
CompletionOfCollegePreparatoryProgram STRING,
Recommendations STRING,
FormalDemonstrationOfCompetencies STRING,
```

```
AdmissionTestScores STRING,
Toefl STRING,
OtherTest STRING,
ApplicantsTotal STRING,
ApplicantsMen STRING,
ApplicantsWomen STRING,
AdmissionsTotal STRING,
AdmissionsMen STRING,
AdmissionsWomen STRING,
EnrolledTotal STRING,
EnrolledMen STRING,
EnrolledWomen STRING,
EnrolledFullTimeTotal STRING,
EnrolledFullTimeMen STRING,
EnrolledFullTimeWomen STRING,
EnrolledPartTimeTotal STRING,
EnrolledPartTimeMen STRING,
EnrolledPartTimeWomen STRING,
FirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores
        STRING,
PercentFirstTimeDegreeOrCertificateSeekingStudentsSubmittingSATScores
        STRING,
FirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores
        STRING,
PercentOfFirstTimeDegreeOrCertificateSeekingStudentsSubmittingACTScores
        STRING,
SATEvidenceBasedReadingWriting25thPercentileScore STRING,
SATEvidenceBasedReadingWriting75thPercentileScore STRING,
SATMath25thPercentileScore STRING,
SATMath75thPercentileScore STRING,
ACTComposite25thPercentileScore STRING,
ACTComposite75thPercentileScore STRING,
ACTEnglish25thPercentileScore STRING,
ACTEnglish75thPercentileScore STRING,
ACTMath25thPercentileScore STRING,
ACTMath75thPercentileScore STRING
)
;
```

11.5 Admission Statistics Load data

The service layer/fact table **AdmissionStat** is loaded from the OD table **od_AdmissionStat** by year. Every time this fact table is updated, it's loaded with the latest data for that year. We need to take enough precautions to ensure the quality of data being loaded in the ETL process. Therefore, we have to delete old data loaded for that year and new data is loaded from the OD table. We've created a stored procedure **pr_AdmissionStat_Load** using the Snowflake. Inside the JavaScript, we have used 2 variables, **sql_DelCommand** and **sql_Command**. The variable sql_DelCommand contains the SQL commands to delete the records from the fact table for the year passed to the stored procedure. The other variable sql_Command contains the main logic for updating the fact table for that year.

We've used a **LEFT OUTER JOIN** with AcademicInstitution on InstitutionIdentifier to get the matching surrogate key from the dimension table AcademicInstitution. In any case, if we couldn't find the matching record, we are updating the AcademicInstitutionUniqueDWSID in the fact table with -1. To achieve this, we have used Snowflake **IFNULL** command as **IFNULL(D.AcademicInstitutionUniqueDWSID, - 1)**

Here is the Snowflake stored procedure for processing the AdmissionStat table.

```
CALL pr_AdmissionStat_Load(2017::FLOAT);
CALL pr_AdmissionStat_Load(2018::FLOAT);
CALL pr_AdmissionStat_Load(2019::FLOAT);
```

11.6 Summary

In this chapter, we have gone through the data load process for the Parquet files in Snowflake. Data processing architecture for the Parquet file format, reamins very similar to CSV; Snowflake makes it easy to consume this data. The admission statistics table is used by various institutions. We loaded 3 years of admission statistics in the stage file. From the staged file, we created and loaded the OD table and later, the fact table for the servicing layer.

> ***Note***
>
> Code available to download at
>
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/source/code/SQL/AdmissionStat

12 Semi-structured data Load

12.1 Lookup entities

Files generated at the source system contains code value pair records for multiple lookup tables. Sample data from the source system is shown in the below table.

Source system data layout

CodeName	TableData
stabbr	'AL'='Alabama'
stabbr	'AK'='Alaska'
stabbr	'AZ'='Arizona'
ACT	'A'='Active - institution active'
ACT	'N'='New (active) - added during the current year'
ACT	'R'='Restore (active) - restored to the current universe'

The first column helps to identify the target entity and the second column is actual data row for the target tables. Data for multiple columns is separated by an **'='** sign.

In the above example

The StateAbbrevation table should be loaded with two attributes. The attribute values for the first row are **'AL'** and **'Alabama'**. Similarly AccountStatus should be loaded with the the values **'A'** and **'Active - institution active'**

Data is available for processing in the staged file **@IPEDS_CM**.

The following query are used to get rows from staged file

```
SELECT  t.$1 AS CodeName,
               t.$2  TableData
FROM   @IPEDS_CM t
```

We have added a mapping table **CodeDestinationTable** that provides destination table name for the code.

The following script will create and load the CodeDestinationTable table.

```
CREATE TABLE CodeDestinationTable
(
                    Code STRING,
                    TableName STRING
);


INSERT INTO CodeDestinationTable (Code,TableName)
SELECT 'ACT','InstitutionStatus'  UNION ALL
SELECT 'c15basic','CarnegieClassification2015'  UNION ALL
SELECT 'c18basic','CarnegieClassification2018'  UNION ALL
SELECT 'c18ipug','CarnegieClassification2000'  UNION ALL
SELECT 'carnegie','CarnegieClassification'  UNION ALL -- Same name as above
SELECT 'cbsa','CoreBasedStatisticalArea'  UNION ALL
```

```
SELECT 'cbsatype','CBSATypeMetropolitan'  UNION ALL
SELECT 'ccbasic','CarnegieClassification2005'  UNION ALL
SELECT 'cngdstcd','StateCongressionalDistricID'  UNION ALL
SELECT 'control','InstitutionControl'  UNION ALL
SELECT 'countycd','CountryCode'  UNION ALL
SELECT 'csa','CombinedStatisticalArea'  UNION ALL
SELECT 'cyactive','InstitutionActive'  UNION ALL
SELECT 'deathyr','InstitutionDeletedYear'  UNION ALL
SELECT 'deggrant','DegreeGrantingStatus'  UNION ALL
SELECT 'dfrcgid','NCESDataFeedbackReport'  UNION ALL
SELECT 'dfrcuscg','InstitutionDataFeedbackReport'  UNION ALL
SELECT 'f1syscod','IdentificationNumberMultiCampusOrganization' UNION ALL
SELECT 'f1systyp','MultiinstitutionMulticampusOrganization' UNION ALL
SELECT 'fips','FIPSStateCode'  UNION ALL
SELECT 'groffer','GraduateOffering'  UNION ALL
SELECT 'hbcu','HistoricallyBlackCollege'  UNION ALL
SELECT 'HDEGOFR1','HighestDegreeOffered'  UNION ALL
SELECT 'hloffer','HighestLevelOffering'  UNION ALL
SELECT 'hospital','HospitalInstitution'  UNION ALL
SELECT 'iclevel','InstitutionLevel'  UNION ALL
SELECT 'instcat','InstitutionalCategory'  UNION ALL
SELECT 'instsize','InstitutionSizeCategory'  UNION ALL
SELECT 'landgrnt','LandGrantInstitution'  UNION ALL
SELECT 'locale','UrbanizationDegree'  UNION ALL
SELECT 'medical','MedicalDegreeGranted'  UNION ALL
SELECT 'necta','NewEnglandArea'  UNION ALL
SELECT 'obereg','EconomicBureauRegion'  UNION ALL
SELECT 'opeflag','TitleIVEligibilityCode'  UNION ALL
SELECT 'openpubl','InstitutionOpenGeneralPublic'  UNION ALL
SELECT 'postsec','PrimaryPostsecondaryIndicator'  UNION ALL
SELECT 'pseflag','PostsecondaryInstitution'  UNION ALL
SELECT 'pset4flg','PostsecondaryTitleIVInstitutionIndicator' UNION ALL
SELECT 'rptmth','ReportingMethod'  UNION ALL
SELECT 'sector','InstitutionSector'  UNION ALL
SELECT 'stabbr','StateAbbreviation'  UNION ALL
SELECT 'tribal','TribalCollege'  UNION ALL
SELECT 'ugoffer','UndergraduateOffering' ;
```

The **CodeDestination** table is used to create the corresponding OD table, and service layer tables; furthermore, it is used for populating data dynamically from the staged file to the OD tables.

We concatenated **od_** to the table name column from the CodeDestination entity. In the **pr_od_CodeMapping_Load** stored procedure we check if the corresponding table exist in the Snowflake. If the table is there, it is truncated and the staged file data for that code is loaded into the specific OD table. If the table is missing then, a new table is added. The newly added table is used to load data.

The combination of **CodeDestinationTable** table and **@IPEDS_CM** staged file is used to populate mulitple operational data stores. Example when the code column content has value **stabbr** then destination table will be **od_StateAbbreviation**. The content of TableData column from **@IPEDS_CM** where **$1 = 'StateAbbreviation'** will be used to populate **od_StateAbbreviation**.

The following script outlines this process.

```
CREATE OR REPLACE PROCEDURE pr_od_CodeMapping_Load()
RETURNS STRING
LANGUAGE javascript
```

```
AS
$$
var sql_GetTables = `SELECT DISTINCT UPPER(s.$1) AS CodeName ,
                        UPPER(TableName) AS DestinationTable
                FROM @IPEDS_CM s
                INNER JOIN CodeDestinationTable
                        ON s.$1 =  CodeDestinationTable.Code
                ORDER BY CodeName;`

CmdTables = {sqlText: sql_GetTables};
StmtTables = snowflake.createStatement(CmdTables);
rsTables = StmtTables.execute();
var strSQLCommands = '';
var strTablePrefix = ` od_`
while (rsTables.next())  {
var strStgTableName         = strTablePrefix
        + rsTables.getColumnValue("DESTINATIONTABLE")
var sqlStgTableTruncate     = `TRUNCATE TABLE IF EXISTS `
        + strStgTableName +`;`
var sqlStgTableCreate       = `CREATE TABLE IF NOT EXISTS`
        + strStgTableName+` (Code STRING, Description STRING);`
var sqlTargetTableCreate    = `CREATE TABLE IF NOT EXISTS`
        +' '+ rsTables.getColumnValue("DESTINATIONTABLE")
        +` (Code STRING, Description STRING);`
var sqlStgTableInsert       = `INSERT INTO
        `+ strStgTableName + `(Code,Description)
        SELECT
         LEFT(CodeValuePair,POSITION('=',CodeValuePair,1)-1) Code,
         RIGHT(CodeValuePair,
                LEN(CodeValuePair) - POSITION('=',CodeValuePair,1))
                        TableCodeValue
        FROM(
                SELECT
                        $1 AS CodeName,
                        $2 AS TableData,
                        REPLACE($2,'''','') AS CodeValuePair
                 FROM @IPEDS_CM t
                WHERE UPPER($1) = '
                        `+ rsTables.getColumnValue("CODENAME") + `' ) A;`
var arrSqlCmd =[4];
arrSqlCmd[0] = sqlStgTableTruncate
arrSqlCmd[1] = sqlStgTableCreate
arrSqlCmd[2] = sqlTargetTableCreate
arrSqlCmd[3] = sqlStgTableInsert
var sql_ExecCommand ='';
for (var j=0;j<arrSqlCmd.length; j++)
{
        sql_ExecCommand = arrSqlCmd[j]
        try
        {
                snowflake.execute(
                        {sqlText: sql_ExecCommand}
                );
        }
        catch(err)
```

```
        {
                return "Failed :" + err;
        }
}
}
return "Success";
$$;
```

On executing the above stored procedure, following tables are added and loaded with data from single staged file.

- od_InstitutionStatus
- od_CoreBasedStatisticalArea
- od_CBSATypeMetropolitan
- od_CarnegieClassification2005
- od_StateCongressionalDistricID
- od_InstitutionControl
- od_CountryCode
- **.....**
- od_IdentificationNumberMultiCampusOrganization
- od_MultiinstitutionMulticampusOrganization
- od_FIPSStateCode
- od_GraduateOffering
- od_NewEnglandArea
- od_EconomicBureauRegion
- od_TitleIVEligibilityCode
- od_InstitutionOpenGeneralPublic

12.2 Merge data

Once data is loaded into the operational data zone, we use a similar approach to merge data to the final servicing layer entities. We check underlying entities dynamically, any missing entities are added. Data is merged from the OD to these entities using **pr_CodeMapping_Load**

```
DROP PROCEDURE IF EXISTS pr_CodeMapping_Load();

CREATE OR REPLACE PROCEDURE pr_CodeMapping_Load()
RETURNS STRING
LANGUAGE javascript
AS
$$
        var sql_GetTables =
        `SELECT DISTINCT TableName AS DestinationTable
      FROM CodeDestinationTable
      ORDER BY DestinationTable;`

        CmdTables = {sqlText: sql_GetTables};
```

```
        StmtTables = snowflake.createStatement(CmdTables);
        rsTables = StmtTables.execute();
        //var strCommands = '';
        var strTablePrefix = `od_`
        while (rsTables.next())  {
        var strStgTableName             = " "
                        + strTablePrefix
                        + rsTables.getColumnValue("DESTINATIONTABLE")
        var sqlMergeSQL           =
        `MERGE INTO `
                + rsTables.getColumnValue("DESTINATIONTABLE") + ` t
           USING ` + strStgTableName + ` s ON t.Code = s.Code
                WHEN MATCHED AND (t.Description <> s.Description) THEN
                        UPDATE SET Description  = s.Description
                WHEN NOT MATCHED THEN
                        INSERT (Code,Description)
                        VALUES (s.Code,s.Description)
                        ;`
                //strCommands = strCommands + sqlMergeSQL +'-->'
                try
                {
                        snowflake.execute(
                           {sqlText: sqlMergeSQL}
                        );
                }
                catch(err)
                {
                        return "Failed :" + err;
                }
        }
        return "success" //strCommands
$$;
```

The complete loading of code mapping data is achieved by calling these stored procedures as below.

```
CALL  pr_od_CodeMapping_Load();
CALL  pr_CodeMapping_Load();
```

12.3 Summary

There is lot of flexibility built in Snowflake to manage variety of data needs. We learned, how to load data from the single file into the multiple entities with different set of attributes. With JavaScript based stored procedures, we can consume variety of data layout programmatically.

> ***Note***
>
> Code available to download at
>
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/source/code/SQL/CodeMapping

13 Pipeline Orchestration

13.1 Workflow

In this chapter, we are going to administar all the Snowflake transformation through single point of entry without using any third party tools.

The steps laid out here, are easy to navigate, provide full control over sequencing, and can be easily extended to accommodate other needs as per different requirements.

We have added a workflow table. Attributes on this table are easy to understand for the end users.

- StageName - Description of the pipeline
- ProcedureName - Name of stored procedure to be called
- SourceTableName - Input table used in Snowflake data transformation
- TargetTableName - Destination table for data used in the above procedure
- ExecutionOrder - What is sequence for this step to be executed

The following script is used to create the **IPEDS_Workflow** table

```
CREATE OR REPLACE TABLE IPEDS_Workflow
( StageName STRING,
  ProcedureName STRING,
  SourceTableName STRING,
  TargetTableName STRING,
  ExecutionOrder NUMBER
);
```

We can easily enhance the framework from this chapter with additional functionality, like actions on success and failure and parallel processing. The results of workflow steps are stored in the **IPEDS_Audit** table. The IPDS_Audit table holds information around steps executed, including:

- row count from the source table
- row count before the process in the destination table
- row count after the step execution in the destination table
- date and time when the step finishes execution

```
CREATE OR REPLACE TABLE IPEDS_Audit
( StageName STRING,
  ProcedureName STRING,
  SourceTableName STRING,
  SourceRowCount NUMBER,
  TargetTableName STRING,
  PreRowCount NUMBER,
  PostRowCount NUMBER,
  ProcessDateTime DATETIME default current_timestamp
);
```

The following script is used to populate **IPEDS_Workflow**. The script list the Stored procedure (SP) getting called, the input table name, the output table name, and sequence number to execute this SP.

```
TRUNCATE TABLE IPEDS_Workflow;

INSERT INTO IPEDS_Workflow (
        StageName
        ,ProcedureName
        ,SourceTableName
        ,TargetTableName
        ,ExecutionOrder
        )
SELECT 'OD_TRUNCATE' AS StageName
        ,'pr_od_Truncate()' AS ProcedureName
        ,'' AS SourceTableName
        ,'' TargetTableName
        ,0 ExecutionOrder

UNION ALL

SELECT 'Od_HDR' AS StageName
        ,'pr_od_AcademicInstitution_Load(@Year)' AS ProcedureName
        ,'' AS SourceTableName
        ,'od_AcademicInstitution' TargetTableName
        ,1 ExecutionOrder

UNION ALL

SELECT 'SL_HDR' AS StageName
        ,'pr_AcademicInstitution_Load(@Year)' AS ProcedureName
        ,'od_AcademicInstitution' AS SourceTableName
        ,'AcademicInstitution' TargetTableName
        ,2 ExecutionOrder

UNION ALL

SELECT 'Od_EFFY' AS StageName
        ,'pr_od_Enrollment_Load(@Year)' AS ProcedureName
        ,'' AS SourceTableName
        ,'od_Enrollment' TargetTableName
        ,3 ExecutionOrder

UNION ALL

SELECT 'SL_EFFY' AS StageName
        ,'pr_Enrollment_Load(@Year)' AS ProcedureName
        ,'od_Enrollment' AS SourceTableName
        ,'Enrollment' TargetTableName
        ,4 ExecutionOrder

UNION ALL

SELECT 'OD_IC' AS StageName
        ,'pr_od_InstitutionalCharge_Load(@Year)' AS ProcedureName
        ,'' AS SourceTableName
        ,'od_InstitutionalCharge' TargetTableName
        ,5 ExecutionOrder
```

```
UNION ALL

SELECT 'SL_IC' AS StageName
        ,'pr_InstitutionalCharge_Load(@Year)' AS ProcedureName
        ,'od_InstitutionalCharge' AS SourceTableName
        ,'InstitutionalCharge' TargetTableName
        ,6 ExecutionOrder

UNION ALL

SELECT 'SL_ICBranch' AS StageName
        ,'pr_InstitutionalChargeByAcademicBranch_Load(@Year)' AS
        ProcedureName
        ,'' AS SourceTableName
        ,'InstitutionalChargeByAcademicBranch' TargetTableName
        ,7 ExecutionOrder

UNION ALL

SELECT 'SL_ICPublication' AS StageName
        ,'pr_InstitutionalChargeByPublication_Load(@Year)' AS ProcedureName
        ,'' AS SourceTableName
        ,'InstitutionalChargeByPublication' TargetTableName
        ,8 ExecutionOrder

UNION ALL

SELECT 'SL_ICCategory' AS StageName
        ,'pr_InstitutionalChargeByCategory_Load(@Year)' AS ProcedureName
        ,'' AS SourceTableName
        ,'InstitutionalChargeByCategory' TargetTableName
        ,9 ExecutionOrder

UNION ALL

SELECT 'OD_ADM' AS StageName
        ,'pr_od_AdmissionStat_Load(@Year)' AS ProcedureName
        ,'' AS SourceTableName
        ,'od_AdmissionStat' TargetTableName
        ,10 ExecutionOrder

UNION ALL

SELECT 'SL_ADM' AS StageName
        ,'pr_AdmissionStat_Load(@Year)' AS ProcedureName
        ,'od_AdmissionStat' AS SourceTableName
        ,'AdmissionStat' TargetTableName
        ,11 ExecutionOrder

UNION ALL

SELECT 'OD_CM' AS StageName
        ,'pr_od_CodeMapping_Load()' AS ProcedureName
        ,'' AS SourceTableName
        ,'' TargetTableName
```

```
        ,12 ExecutionOrder

UNION ALL

SELECT 'SL_CM' AS StageName
        ,'pr_CodeMapping_Load()' AS ProcedureName
        ,'' AS SourceTableName
        ,'' TargetTableName
        ,13 ExecutionOrder;
```

The following code can help to understand the steps that will be executed

```
SELECT * FROM IPEDS_Workflow ORDER BY ExecutionOrder;
```

The stored procedure **pr_IPEDS_Process** is used as a master controller. When we call this controller the following steps are executed.

- All data from the **IPEDS_Workflow** table is read in order of ExecutionOrder
- For each of the rows
- It will record the row count from the source and destination table.
- It will execute the stored procedure in the ProcedureName attribute
- It will record the row count on the destination table.

```
CREATE
        OR REPLACE PROCEDURE pr_IPEDS_Process (YEAR FLOAT)
RETURNS STRING LANGUAGE javascript AS $$
        var sql_WF = `SELECT StageName
        ,ProcedureName
        ,SourceTableName
        ,TargetTableName
        ,ExecutionOrder
FROM IPEDS_Workflow
ORDER BY ExecutionOrder`;

CmdSql =

{sqlText: sql_WF};

StmtSql = snowflake.createStatement(CmdSql);

rsSql = StmtSql.

EXECUTE ();

try{

WHILE (rsSql.NEXT ()) { var strSourceTable = rsSql.
        getColumnValue("SOURCETABLENAME");
        var strTargetTable =
                rsSql.getColumnValue("TARGETTABLENAME");

var strProcedureName = '';
```

```
var sourceCount = preCount = postCount = 'NULL';

var sqlSourceRowCount = SqlRowCount = stmtRowCount = '';

var rsPreRowCount
        ,rsPostRowCount;

//Query = Query + '--Source:'
        + strSourceTable + ' Target: ' + strTargetTable;
IF (strSourceTable != '')
        { sqlSourceRowCount = `SELECT Count(1) AS
        Counts FROM ` + strSourceTable +
                        ` WHERE AcademicYear = ` + YEAR.toString() + `;
        `;
var stmtSourceRowCount
        = snowflake.createStatement({sqlText:
                sqlSourceRowCount});

var rsSourceRowCount = stmtSourceRowCount.

EXECUTE ();

IF (rsSourceRowCount.NEXT ()) { sourceCount =
        rsSourceRowCount.getColumnValue("COUNTS");
        } }

// Get Pre rowcount if the target table is provided.
IF (strTargetTable != '') { SqlRowCount =
        `SELECT COUNT(1) AS Counts FROM `
         + strTargetTable + ` WHERE AcademicYear = `
         + YEAR.toString() + `;
        `;

//Query = Query + '--Target:' + SqlRowCount;
stmtRowCount =
        snowflake.createStatement({sqlText: SqlRowCount});

rsPreRowCount = stmtRowCount.

EXECUTE ();

IF (rsPreRowCount.NEXT ()) { preCount = rsPreRowCount.
        getColumnValue("COUNTS");
        } }
        strProcedureName = `CALL `
                + rsSql.getColumnValue("PROCEDURENAME").
        replace('@Year', YEAR.toString()) + `;

`;

//Query = Query + '--Proc:' + strProcedureName;
var stmtProcessData = snowflake.createStatement({sqlText:
                strProcedureName});
```

```
stmtProcessData.

EXECUTE ();

// Get Post rowcount if the target table is provided.
IF (SqlRowCount != '') { rsPostRowCount = stmtRowCount.
        EXECUTE ();

IF (rsPostRowCount.NEXT ()) { postCount = rsPostRowCount
        .getColumnValue("COUNTS") } }
 var sql_Audit =
        `INSERT INTO IPEDS_Audit(StageName, ProcedureName,
        SourceTableName, SourceRowCount, TargetTableName, PreRowCount,
        PostRowCount) VALUES (
        '` + rsSql.getColumnValue("STAGENAME") + `'
        ,'`+ strProcedureName
                  +`'
        ,'`+ strSourceTable +`'
        ,` + sourceCount + `
        ,'`+ strTargetTable + `'
        ,` + preCount + `
        ,` + postCount + `
        );
`;

//Query = Query + '--Audit:' + sql_Audit;
var stmt = snowflake.createStatement({sqlText: sql_Audit});

stmt.

EXECUTE ();

} // End While.
        } catch(err) {

RETURN "Failed:" + err.toString();

}

RETURN 'Success';$$;
```

These row counts, with additional information are written to IPEDS_Audit.

The stored procedure **pr_od_Truncate** is written to truncate all the OD tables before we load data. Code for the stored procedure **pr_od_Truncate** is given below.

```
CREATE OR REPLACE PROCEDURE pr_od_Truncate()
RETURNS STRING
LANGUAGE javascript
AS
$$

var arrTableName =[4];
    arrTableName[0] = 'od_AcademicInstitution'
    arrTableName[1] = 'od_Enrollment'
```

```
    arrTableName[2] = 'od_InstitutionalCharge'
    arrTableName[3] = 'od_AdmissionStat'


    var tableName ='';
    for (var j=0;j<arrTableName.length; j++)
    {
        tableName = arrTableName[j]
        try
        {
            snowflake.execute(
                {sqlText: `TRUNCATE TABLE `+tableName +`;`}
            );
      }
        catch(err)
        {
            return "Failed :" + err;
        }
    }
    return "success"
$$;
```

The combination of truncate OD table and load data from IPEDS dataset in the staging files is performed. with the following code

```
CALL pr_od_Truncate();
CALL pr_IPEDS_Process(2017::FLOAT);
```

13.2 Summary

In this chapter we learned how to build workflow management tool using Snowflake. This process helps to navigate through data lineage across staged files, operational data tables, and servicing layer entities. The process outlined in this chapter will work for the cases where we have single source tables and moving to single destination table. We can easily enhance this to accommodate data lineage tracking for multiple input to one destination , and one input to mulitple destination.

> ***Note***
>
> Code available to download at
>
> https://github.com/versatiledp/ExperimentsSnowflake/tree/main/source/code/SQL/Workflow