Paper 4184-2020

Better Student Application Reporting: A Slowly Changing Dimension and SAS® Data Integration Studio

Lauren Schoenheit, NC State University; Alexander K. Fantroy, NC State University

ABSTRACT

For a selective university, shaping the incoming freshman class requires current operational data on the student applications that have been received, the admissions decisions that have been made, and the students that have committed in response throughout the admissions cycle. A point-in-time comparison with historical data serves as a benchmark to better understand and anticipate the makeup of the incoming students. To meet these demands, we shifted student application reporting from a Base SAS® program (run on an ad hoc basis about twice a week that created a data set of application information for that day, saved to a drive), to a scheduled SAS® Data Integration Studio job that created a slowly changing dimension of year-over-year daily application information stored as an oracle table in the university's enterprise database. Changing how the data was stored reduced storage space, as only new applications and change records are added to the student application table instead of accumulating data sets with a repeated record for every day the application was in the system. Scheduling a SAS Data Integration Studio job removed the task of running the program from a person's workload, as well as standardized the collection of student application data. Together these changes facilitated more frequent snapshots of student application data as well as increased processing efficiency. A daily summary is loaded to SAS® Visual Analytics as the basis for a daily year-over-year analysis of student application decisions.

INTRODUCTION

Admissions officers and enrollment managers need up to date information on applications as they move through the admissions cycle to manage waitlisted applications and understand the makeup of the incoming class. A comparison to previous years' applications gives insight into how the university should prepare to meet the needs of the incoming class. For example, will course demands be similar to last year, will housing need more or fewer first year student beds.

OLD PROCESS

NC State collects application data through the admission application software, Admission Pros®, which loads data nightly to our PeopleSoft® Student Information System. The previous process to pull application data out of SIS and create reports, required manually editing and running a Base SAS® program. This program created and saved a SAS data set with the current application data and then compared current application data to last year's data from approximately the same date. The data set from the previous year was manually selected by browsing a folder, seen in Figure 1, with all of last year's data and picking the data set dated closest, but not after today's date a year ago.

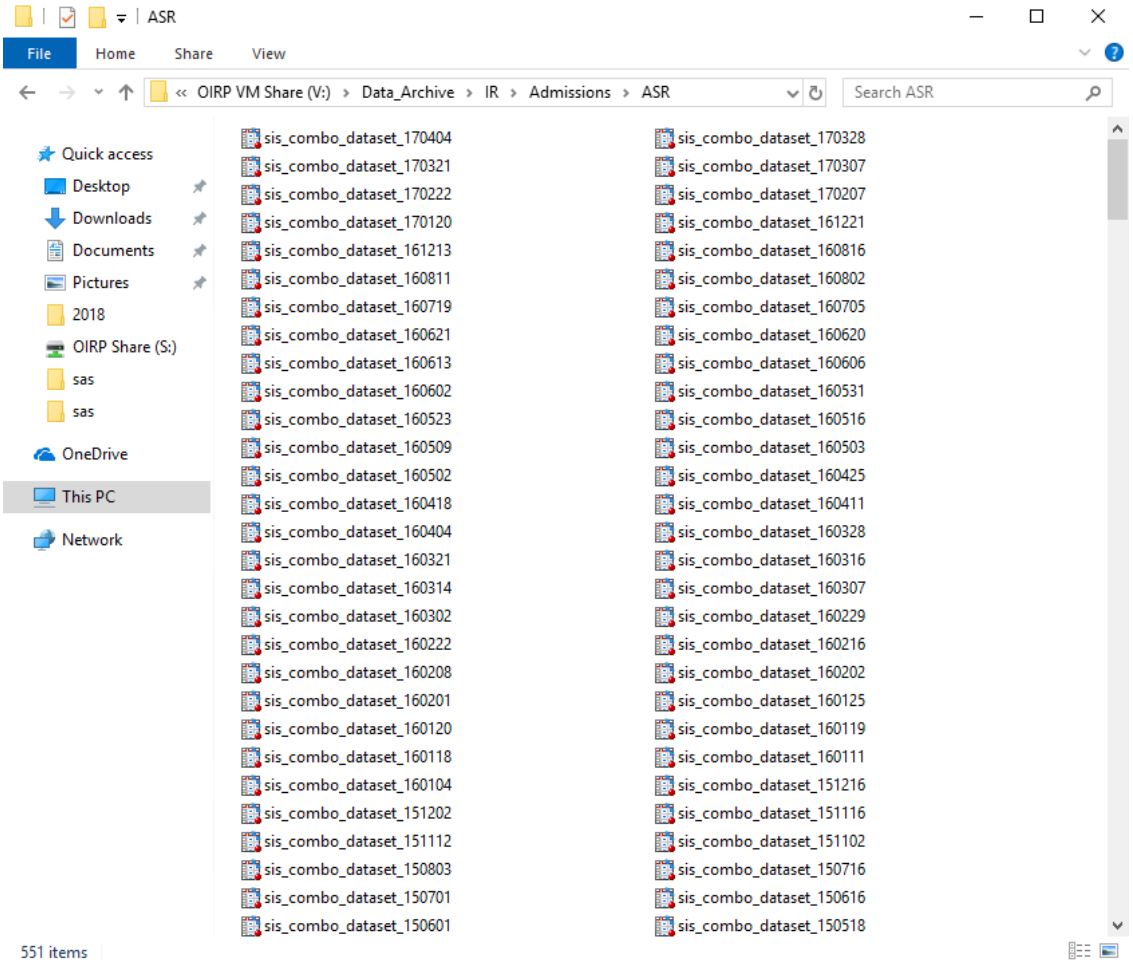


Figure 1. Folder contents

A macro variable in the Base SAS® program was set to the date of the file and today's date; see below:

```
%let ymd   = 160816;  ** YYMMDD <- change for each run to today!! used in file names, not any calcs;

%let lymd6 = 150803;        ** last year yymmdd, must match date in data set name in ASRSIS library; ** in order to make valid comparison;
```

The process was generally run on Tuesdays and Thursdays during the admission cycle, but required someone to be in the office to manually run the program to create the data sets. This caused some weeks to be missed if people were on vacation. If the program had been run inconsistently in the previous year, the comparison between the current application and last year's applications could be up to a two week difference, which can be a substantial amount of time in the admissions cycle. For planning purposes, it is much better to have a consistent comparison between this year and last. Also, at certain times admissions officers would want the data updated more frequently than twice a week, which could not always be accommodated.

In addition to lacking consistent access to the information, the process was inefficient in the time it would take to locate the comparison data set and update the program. The process also required storing a large amount of redundant data, as each data set contained all the applications received for the upcoming fall as of the date the program was run. Two years of application data, the current year and the comparison year required approximately 10 gigs

of space to store. Additionally, the data was stored in SAS data sets that only one office could access. Other offices interested in this data would have to collect and store it independently, causing discrepancies in application data, duplication of work effort, and an even greater waste of university disc space. To address these issues, it was decided to completely abandon the previous process and develop a new process using a slowly changing dimension to create, store, and disseminate application data.

WHAT IS A SLOWLY CHANGING DIMENSION (SCD)?

A slowly changing dimension is a technique to capture the change of your data over time. The dimension table allows you to store current and historical data. We wanted to be able to track all changes of the student's application data while they were in the application process. There are numerous ways to implement a SCD, the most popular are below.

Type 0 – Static

No changes occur to the table

Type 1 – No History

Overwrites current records, which means there are no historical records being stored.

Type 2 – Versioning

Allows rows to be updated by inserting a new record, which keeps a historical entry for the record.

Type 3 – Previous Value

When a tracked attribute is changed it adds a new column with the previous value of that attribute.

Type 4 – History Table

Updates the current record but loads changes to another table.

Type 6 – Hybrid

Mix of 1,2 and 3

The way you choose to implement your SCD should fit your business needs.

WHAT IS A SLOWLY CHANGING DIMENSION (SCD)?

We chose to use the Type 2 method since it checked all the right boxes for our business needs. The 2 main priorities of this project were to be able to track a student's application process and to have a point in time comparison to previous years' applications. With the type 2 implementation we can record the changes to students' applications while also keeping the historical versions of their applications. This gives us the ability to grab point in time records which we then can compare to previous point in time records. If we wanted to compare this year's applications from Jan 1st through March 1st to last years during the same time period, this implementation gives us the ability to do that. It also gives us that version of the student's application and not the most current version.

IMPLEMENTATION

We used SAS® Data Integration Studio to implement our ETL processes. Within DI Studio there are many provided transformations that you can use to help with the manipulation of data. The transformation showed in Figure 2 is used to implement our SCD. One of the most important steps when implementing SCD Type 2 is to decide on which attributes you want the SCD to track, which will trigger the insertion of new records.

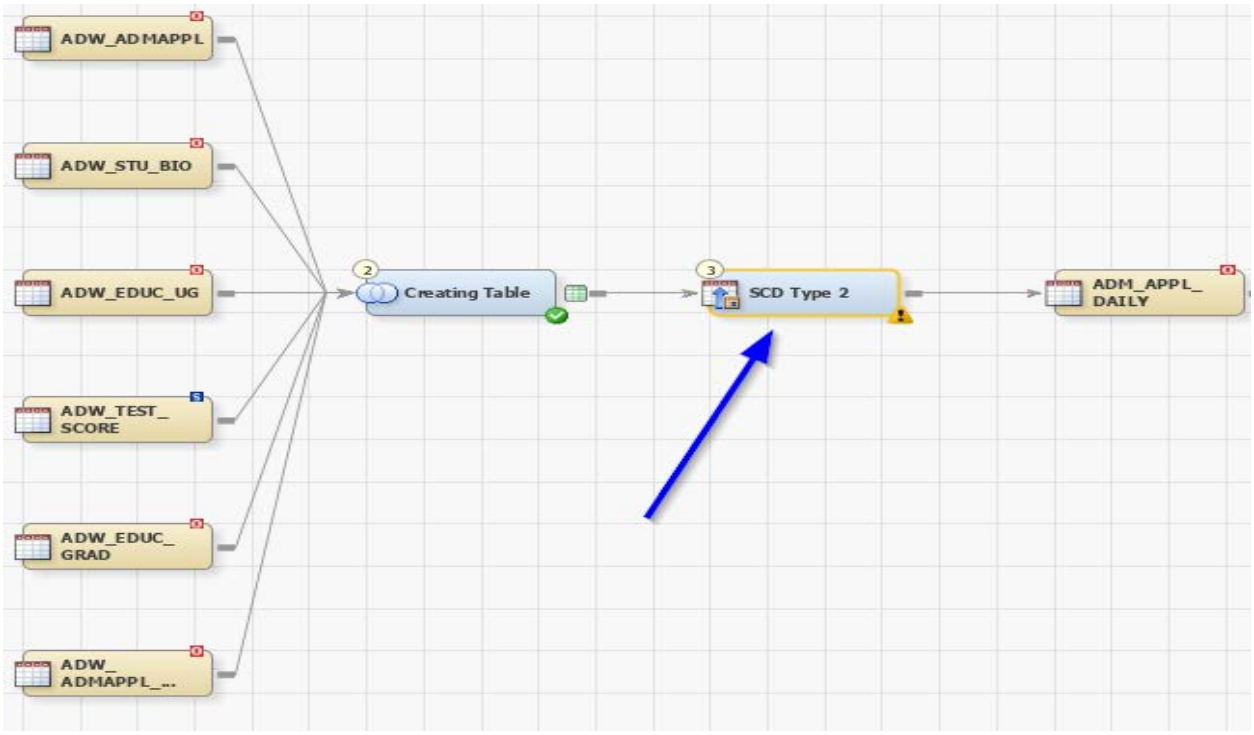


Figure 2. DI Job to implement SCD

The first step is to identify which columns you want to be overwritten with updated information. These are Type 1 columns and we do not need to track their changes, we just want the most updated values (Figure 3).

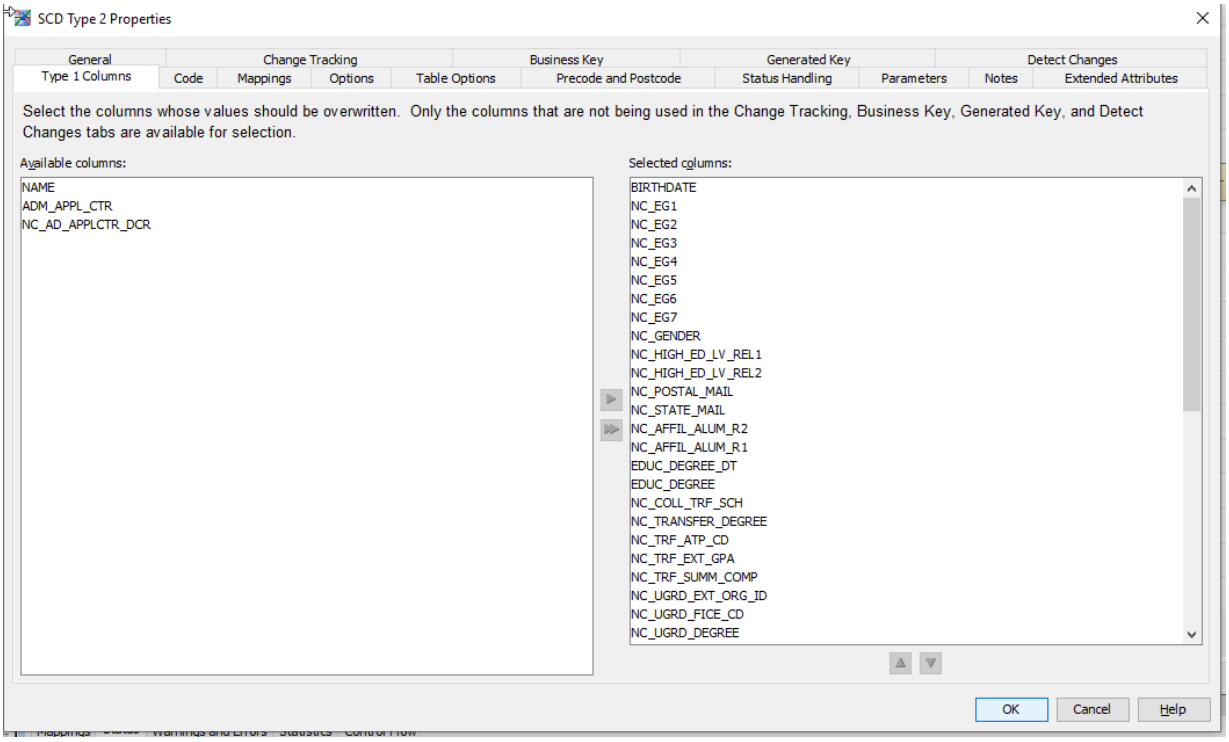


Figure 3. Tables columns

The second step is to identify the columns in your SCD that will store the change indicators when a new record is inserted (Figure 4).

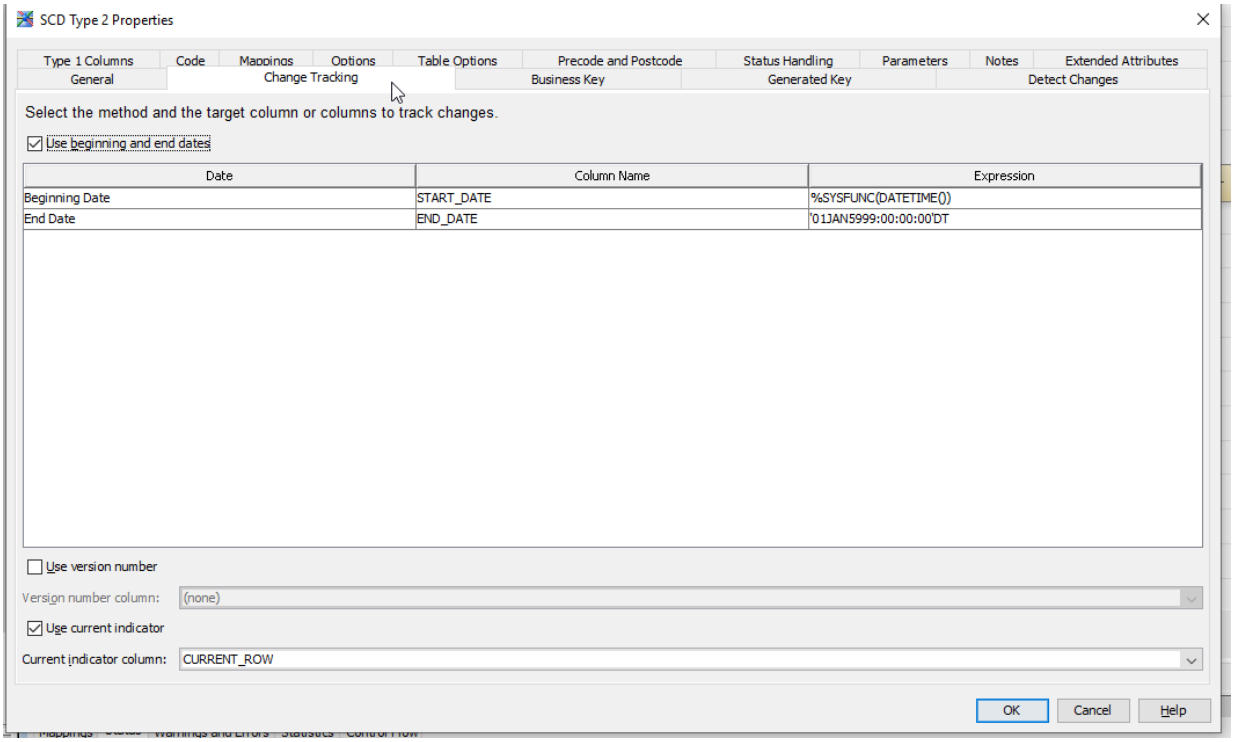


Figure 4. Selected change indicators

The third step is to identify the field where you will be storing the unique key for the SCD. This key is automatically uniquely generated for every record inserted into the SCD (Figure 5).

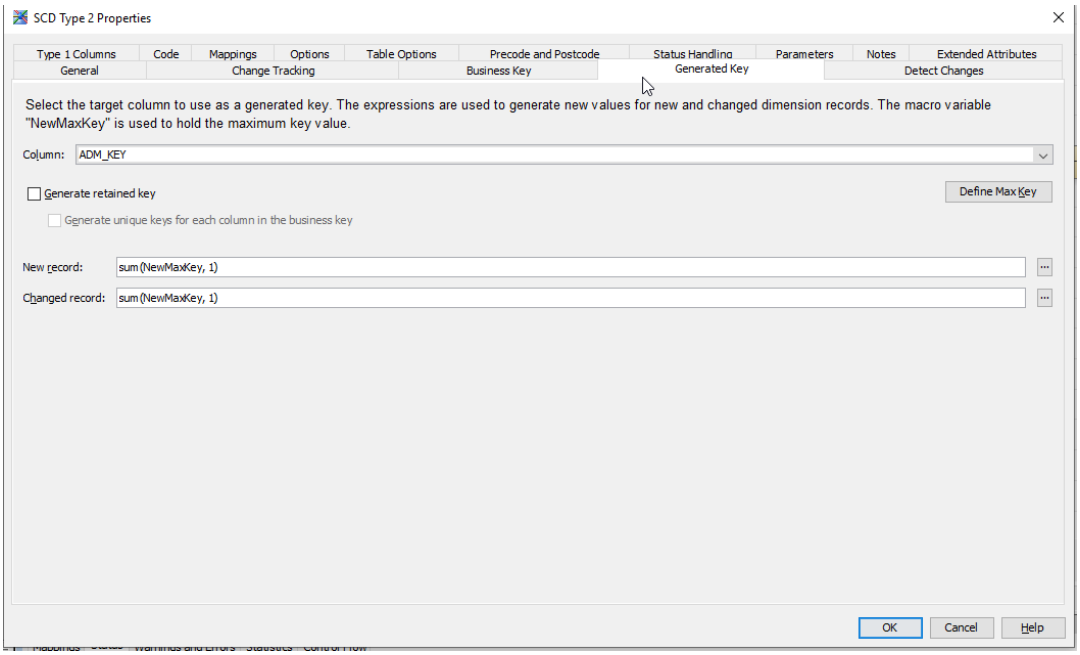


Figure 5. Identify unique key variable

The fourth step is to find the fields that uniquely identify your Business key. For this project the Business key was the student's unique student ID number and the student's unique application number (Figure 6).

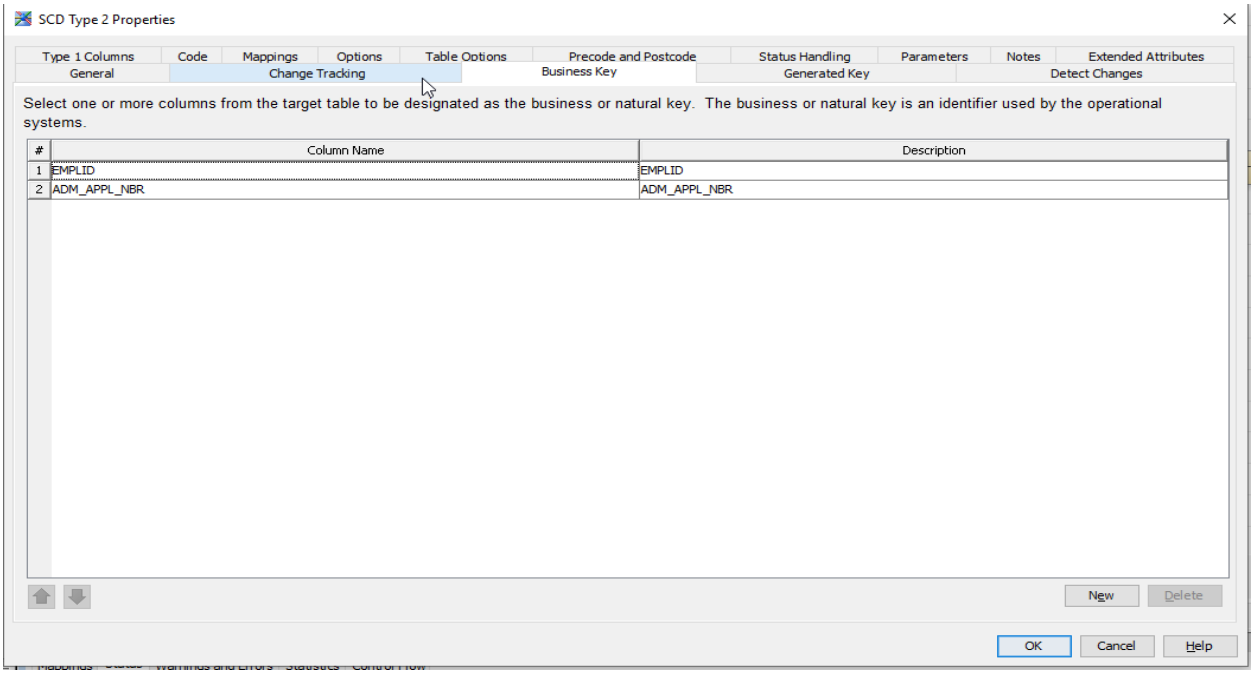


Figure 6. Identify business key

The last step is to identify the columns you want the SCD to track to detect changes. Every time the SCD detects a change in one of these columns, a new entry will be inserted with the same business key, but with updated values of the columns that have changed (Figure 7).

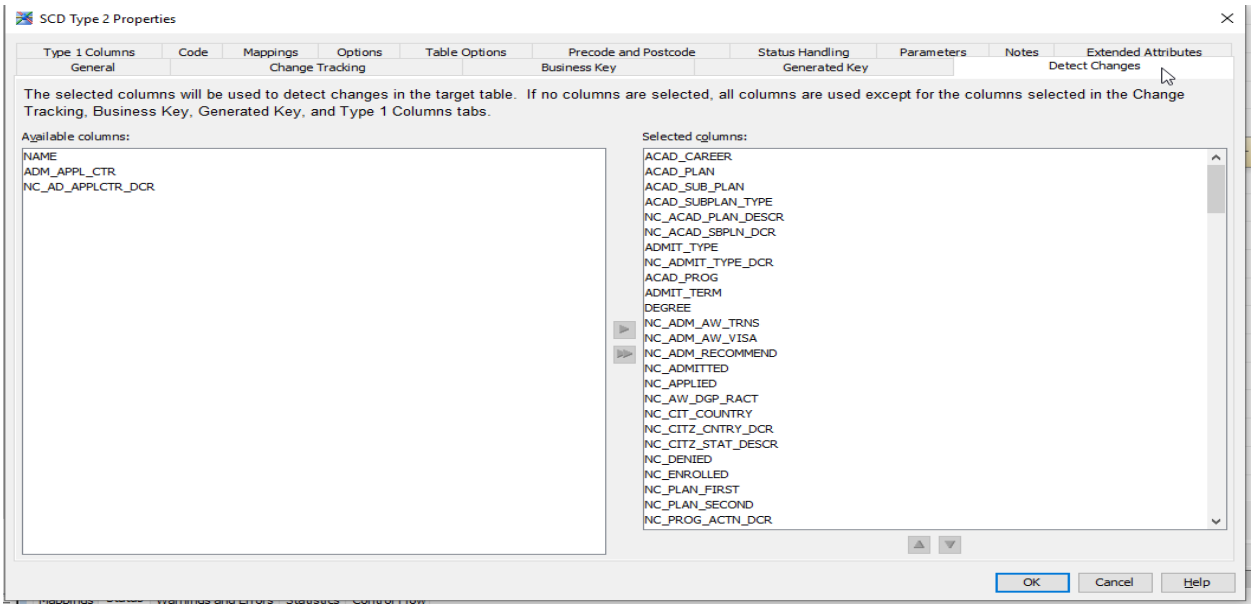


Figure 7. Columns to track changes

Our implementation of the Type 2 SCD will monitor a set of attributes of the student's application and will insert a new record when they are changed. For example, one of the attributes that we selected to monitor was application status (nc_prog_reasn_dcr). Our process runs on a nightly bases to pick up changes to the student's application. For example, if a student has a change to his application status, a new record is inserted into the SCD with a new start date, closing the previous record with a new end date, and setting the current_row = 'Y' for the new record. As you can see in Figure 2, this student's

application status has changed multiple times and all the previous attributes of the student's application are saved in the new row and only the new entry of the application status is updated. The important attributes in the new row are start_date, end_date, current_row and adm_key (Figure 8).

Start Date – The date the value of the attribute was changed.

End Date – The date the record was closed due to a change in one of the attributes that is being tracked.

ADM Key – The unique identifier for the SCD.

Current Row – Indicates the most recent record for that application.

	EMPLID	AD...	ACAD_PLAN	ACAD_CAREER	NC_EG_SUMMARY	RESIDENCY	NC_ADMITTED	NC_APPLIED	ACTCOMP	NC_PROG_REASN_DCR	START_DATE	END_DATE	ACTHMTH	CURRENT_ROW	ADM_KEY
1	200...	005...	16INTSTBA	UGRD	White	OUT	N	Y	(null)	(null)	24-OCT-18	20-NOV-18	(null)	N	1157309
2	200...	005...	16INTSTBA	UGRD	White	OUT	N	Y	(null)	(null)	20-NOV-18	12-DEC-18	(null)	N	1172949
3	200...	005...	16INTSTBA	UGRD	White	OUT	N	Y	30	(null)	12-DEC-18	22-FEB-19	26	N	1191189
4	200...	005...	16INTSTBA	UGRD	White	OUT	Y	Y	30	A01-Admit - Fall	22-FEB-19	16-APR-19	26	N	1246073
5	200...	005...	16INTSTBA	UGRD	White	OUT	Y	Y	30	N01-Withdrawal - Will Not Ente	16-APR-19	01-JAN-99	26	Y	1309786

Figure 8. Example rows from SCD

DATA CLEANING

Now that we have our dimension table/process established we can begin to validate and clean our data. Data is not always in the format that you need and business logic needs to be applied to get clarity out of our data. This process could have been done during the load to the SCD but we wanted to keep that data as raw as possible, just in case we wanted to apply a different set of business logic in the future.

The first step we did was to get the correct people at the table to determine which attributes from the SCD are needed in the reporting data set and which values may need to be transformed. Once that was complete, we began putting together a process to pull the data out of the SCD and load it to the staging area. We did this ETL process also within in DI Studio.

One priority of this project was to create a data set that would give us the ability to do point in time comparisons. We wanted to give Colleges within the University the ability to compare their current application numbers with previous years.

SELECTING REPORTING DATA

The first step was to identify which applications we want to load to the reporting data set and pull them into the staging area. For point in time comparison, we needed all applications submitted up to today and needed all the applications for the same time period for the previous year(s). This point in time data set will consist of 3 sets of data (current year, last year and end of last year). Since our SCD process has been running for multiple years we are able to do this by using start_date, end_date and current_row attributes in the SCD to select the correct records.

Current year = All applications currently submitted in their current state (current_row = 'Y', start_date = current_year)

Last year = All applications submitted at this time last year. (start_date <= todays date of last year) We do not care if it is the most recent version of the application (current_row in 'Y' or 'N') as long as the application falls within the time period. If there are multiple versions of the application during the time period, then we will grab the most recent.

End of last year = The final state of each application at the end of last year's application cycle current_row = 'Y', start_date = last_year. This will give us the last state of this application. This will allow us to see if we are on pace to meet or exceed last year's application numbers.

In the staging area we use the Extract transformation within DI Studio to extract the correct time period and apply necessary business logic (Figure 9).

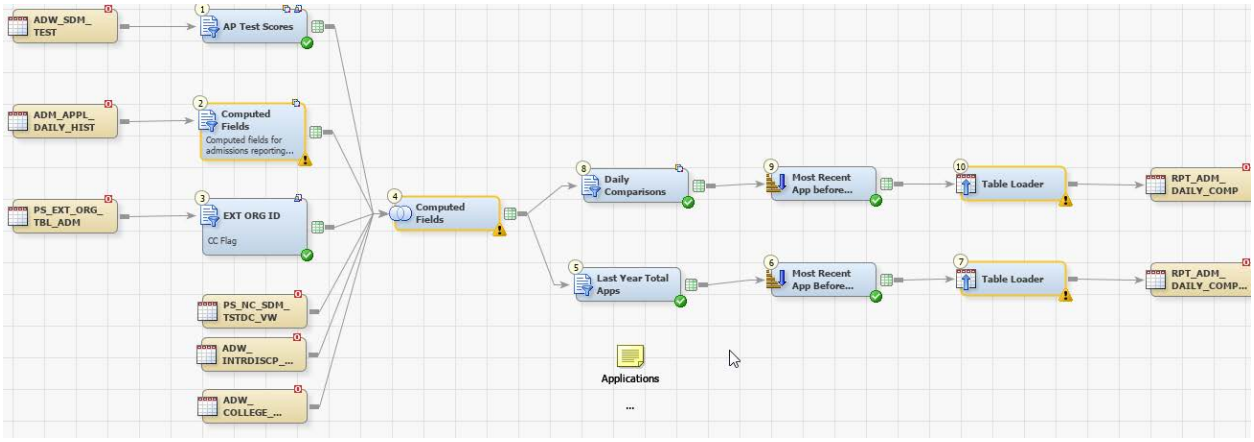
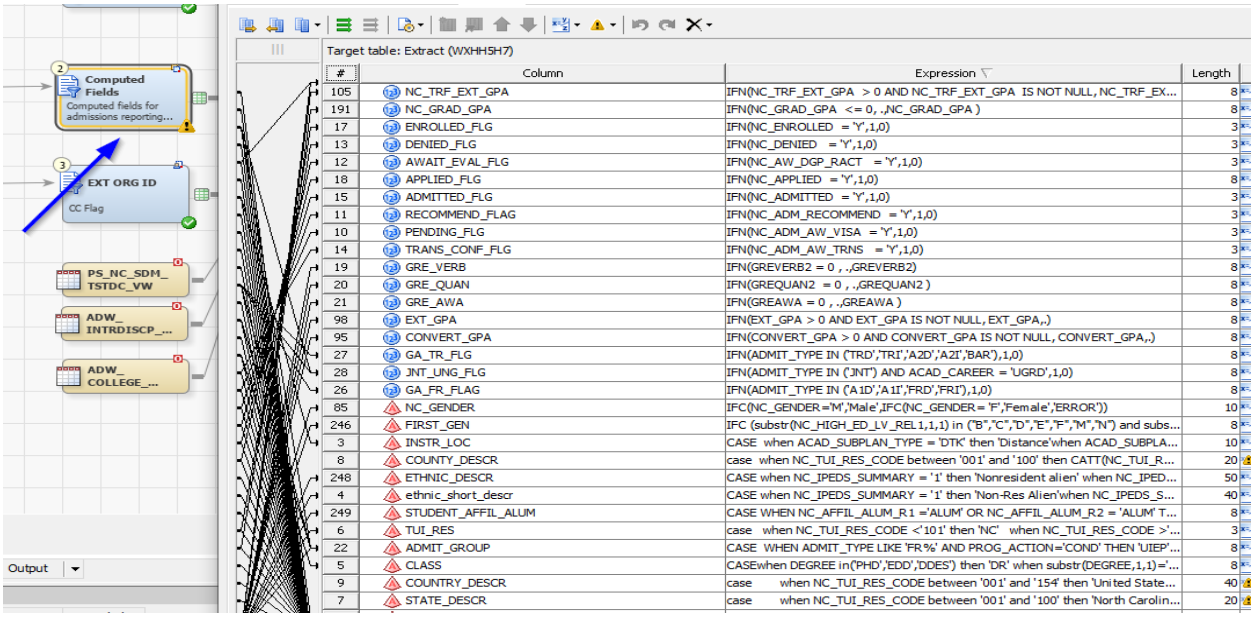


Figure 9. DI job to extract information from SCD

Within the Extract transformation you can see where we begin applying the business logic (Figure 10).



#	Column	Expression	Length
105	NC_TRF_EXT_GPA	IFN(NC_TRF_EXT_GPA > 0 AND NC_TRF_EXT_GPA IS NOT NULL, NC_TRF_EX...	8
191	NC_GRAD_GPA	IFN(NC_GRAD_GPA <= 0, .,NC_GRAD_GPA)	8
17	ENROLLED_FLG	IFN(NC_ENROLLED = 'Y',1,0)	3
13	DENIED_FLG	IFN(NC_DENIED = 'Y',1,0)	3
12	AWAIT_EVAL_FLG	IFN(NC_AW_DGP_RACT = 'Y',1,0)	3
18	APPLIED_FLG	IFN(NC_APPLIED = 'Y',1,0)	8
15	ADMITTED_FLG	IFN(NC_ADMITTED = 'Y',1,0)	3
11	RECOMMEND_FLAG	IFN(NC_ADM_RECOMMEND = 'Y',1,0)	3
10	PENDING_FLG	IFN(NC_ADM_AW_VISA = 'Y',1,0)	3
14	TRANS_CONF_FLG	IFN(NC_ADM_AW_TRNS = 'Y',1,0)	3
19	GRE_VERB	IFN(GREVERB2 = 0 , .,GREVERB2)	8
20	GRE_QUAN	IFN(GREQUAN2 = 0 , .,GREQUAN2)	8
21	GRE_AWA	IFN(GREAWA = 0 , .,GREAWA)	8
98	EXT_GPA	IFN(EXT_GPA > 0 AND EXT_GPA IS NOT NULL, EXT_GPA,.)	8
95	CONVERT_GPA	IFN(CONVERT_GPA > 0 AND CONVERT_GPA IS NOT NULL, CONVERT_GPA,.)	8
27	GA_TR_FLG	IFN(ADMIT_TYPE IN ('TRD','TRI','A2D','A2I','BAR'),1,0)	8
28	JNT_UNG_FLG	IFN(ADMIT_TYPE IN ('JNT') AND ACAD_CAREER = 'UGRD',1,0)	8
26	GA_FR_FLAG	IFN(ADMIT_TYPE IN ('A1D','A1I','FRD','FRI'),1,0)	8
85	NC_GENDER	IFC(NC_GENDER='M','Male',IFC(NC_GENDER= 'F','Female','ERROR'))	10
246	FIRST_GEN	IFC (substr(NC_HIGH_ED_LV_REL1,1,1) in ("B","C","D","E","F","M","N") and subs...	8
3	INSTR_LOC	CASE when ACAD_SUBPLAN_TYPE = 'DTK' then 'Distance'when ACAD_SUBPLA...	10
8	COUNTY_DESCR	case when NC_TUI_RES_CODE between '001' and '100' then CATT(NC_TUI_R...	20
248	ETHNIC_DESCR	CASE when NC_IPEDS_SUMMARY = '1' then 'Nonresident alien' when NC_IPED...	50
4	ethnic_short_descr	CASE when NC_IPEDS_SUMMARY = '1' then 'Non-Res Alien'when NC_IPEDS_S...	40
249	STUDENT_AFFIL_ALUM	CASE WHEN NC_AFFIL_ALUM_R1 ='ALUM' OR NC_AFFIL_ALUM_R2 = 'ALUM' T...	8
6	TUI_RES	case when NC_TUI_RES_CODE <'101' then 'NC' when NC_TUI_RES_CODE >'...	3
22	ADMIT_GROUP	CASE WHEN ADMIT_TYPE LIKE 'FR%' AND PROG_ACTION='COND' THEN 'UIEP'...	8
5	CLASS	CASEwhen DEGREE in('PHD','EDD','DDES') then 'DR' when substr(DEGREE,1,1)='...	8
9	COUNTRY_DESCR	case when NC_TUI_RES_CODE between '001' and '154' then 'United State...	40
7	STATE_DESCR	case when NC_TUI_RES_CODE between '001' and '100' then 'North Carolin...	20

Figure 10. Extraction logic for SCD

I am coder at heart, but DI Studio has decreased my development time for a number of projects. Within these transformations you also have the ability to write as much SAS code as you want. There is a code tab in all the transformations where you can see what SAS code is actually being generated. As you can see in Figure 11, you have the option to modify the code or write your own.

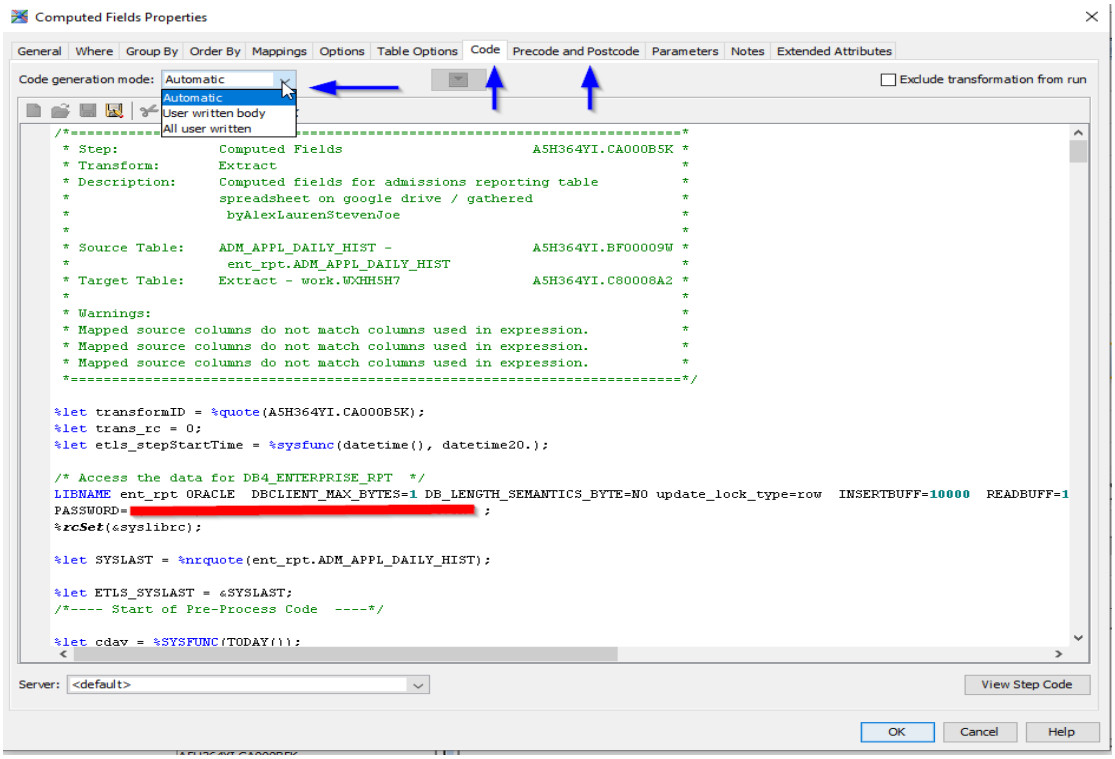


Figure 11. Manually modifying code from DI job

Below is our ETL process flow within DI Studio that is used to create our SCD and reporting data sets (Figure 12).

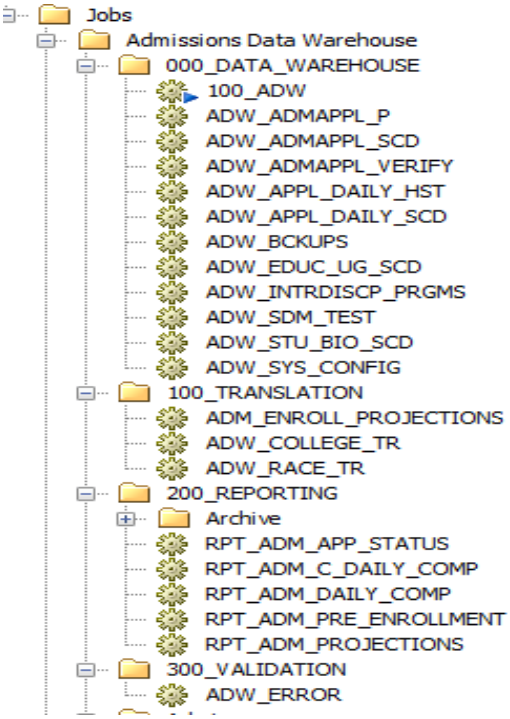


Figure 12. ETL process flow of DI jobs

REPORTING

The last step in the process is the visualization of the data. We currently use SAS® Visual Analytics 7.4 for our reporting needs. The data sets are refreshed/uploaded to the LASR server every morning via DI Studio.

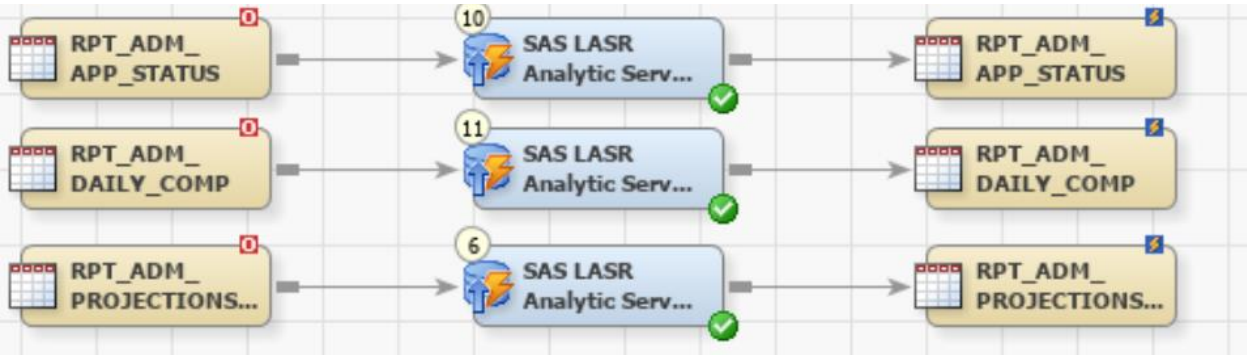


Figure 13. DI Job to load data to LASR server

There are a variety of ways the reports are distributed (email, custom dashboards, saved URL's and embedded reports on the department's website). User access for the reports are controlled through SAS Metadata security. Below are some screenshots of one of the Graduate Schools VA Reports that were created from the final reporting data set. We also have a DEMO version of the report below with artificial data that we use for demonstration purposes that is open to the public. [DEMO Graduate Admissions Management Report]

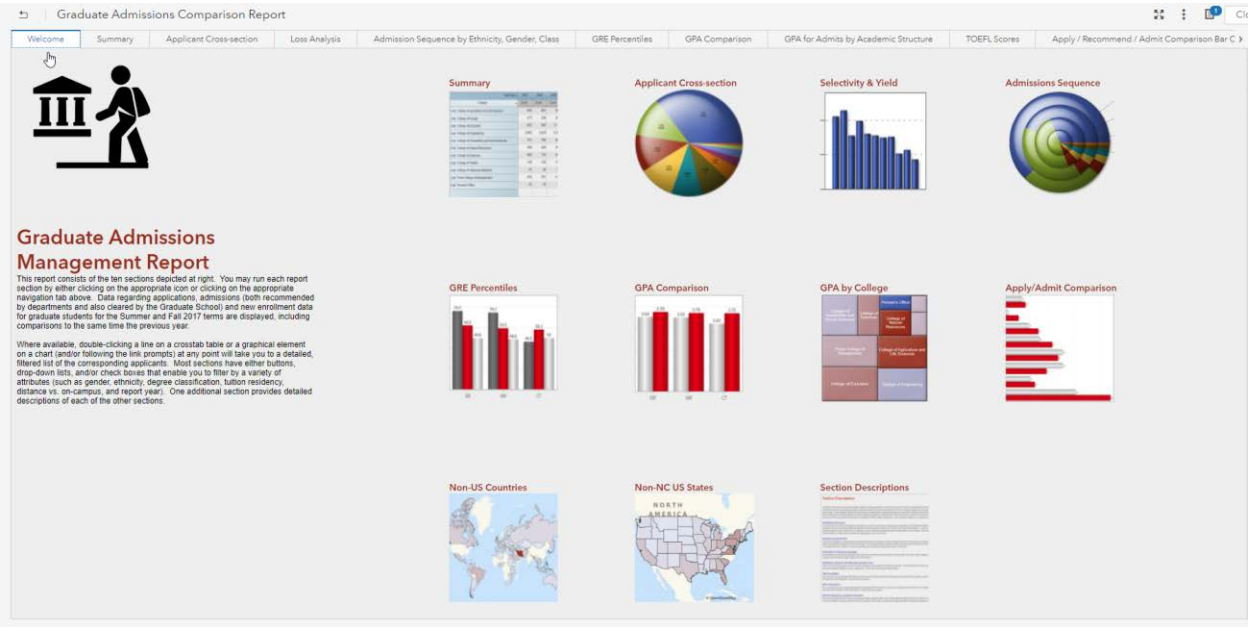


Figure 14. SAS VA admissions data report

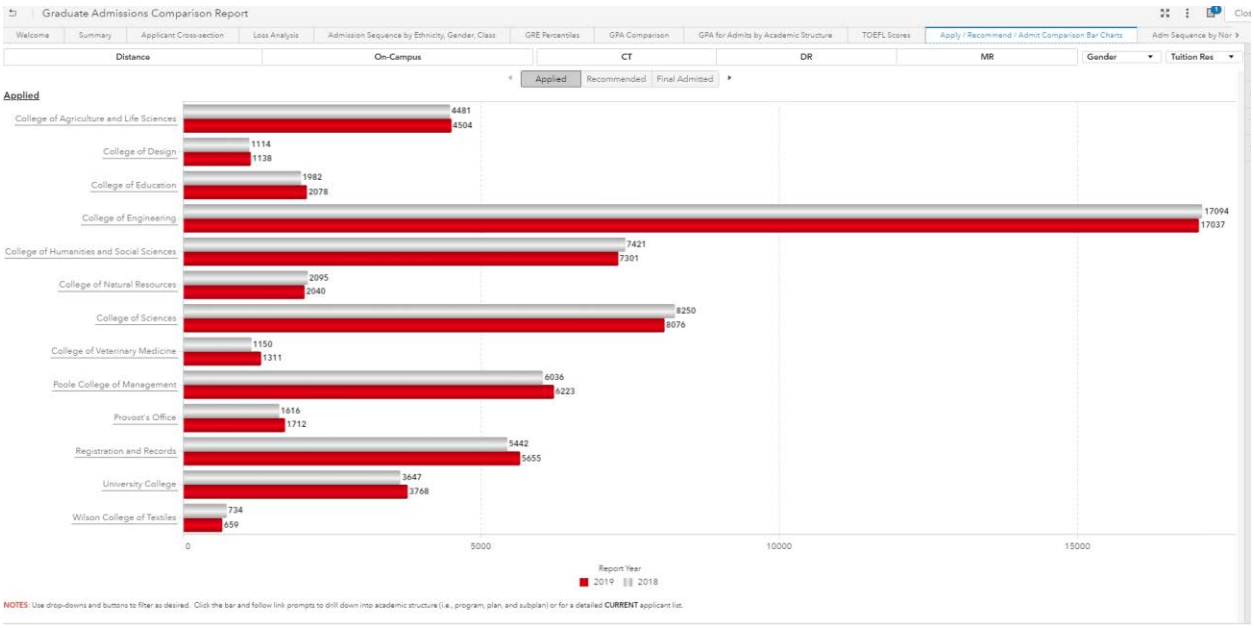


Figure 15. Example report on applications by college

CONCLUSION

This process automated the data capture and creation of the reporting data set, allowing for data to be consistently collected on a daily basis. The process also centralized the data so departments across the university can report admissions data from the same data source saving space and ensuring consistent reporting. Uploading the data to VA nightly refreshes the report giving enrollment managers access to the most recent data.

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

Alexander Fantroy
North Carolina State University
akfantro@ncsu.edu

Lauren Schoenheit
North Carolina State University
leschoen@ncsu.edu