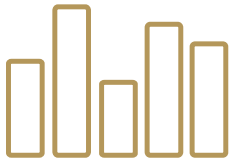


STRATEGIC DATA PROJECT

CLEAN: DATA BUILDING TASKS FOR COLLEGE-GOING

SDP TOOLKIT

FOR EFFECTIVE DATA USE IN EDUCATION AGENCIES

www.gse.harvard.edu/sdp/toolkit

Toolkit Documents

VERSION: 1.1
Last Modified: December 22nd, 2014

Authored by Todd Kawakita and the SDP Research Team

2. Clean: Data Building Guide

Clean and process data files you identify;

Clean: Data Building Guide is a series of tasks with step-by-step instructions to build clean data files from raw data you identify. In each task, analysts take a raw input file and produce a cleaned output file.

Raw input file → Task → Cleaned output file

Each raw input file consists of the data elements required to produce a cleaned output file that matches the specifications for a **research file** in **Identify**: Data Specification Guide.

DATA BUILDING GUIDE

Congratulations on identifying the data to conduct rigorous analyses via **Identify**. **Clean** is the next stage in the **SDP Toolkit**. After this stage, you will have clean data files that allow you to **Connect** and **Analyze** data related to college-going success in your agency.

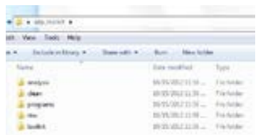
The Tasks

Clean consist of seven tasks:

1. **Task 1**: Student Attributes
2. **Task 2**: Student School Year
3. **Task 3**: Identifying the Ninth Grade Cohort
4. **Task 4**: Student School Enrollment
5. **Task 5**: Student Test Scores
6. **Task 6**: Student Class Enrollment
7. **Task 7**: NSC (National Student Clearinghouse) Data

Each task uses a raw input file and produces a cleaned output file that matches Identify.

Download these raw input files along with everything else you need for the toolkit as a **zip file** at **www.gse.harvard.edu/sdp/toolkit**. When unzipped, this file will reveal an infrastructure including all the steps of the toolkit, the data files you need, and template files that serve as a shell for Stata code.



In particular, in Clean, you will be working with the files in the **raw** folder. If you are using Stata, you can fill in the corresponding do file templates in **programs** to go through the tasks.

How to Start

The beginning of the Data Building Guide is a **Decision Rules Glossary** (p. 6). This glossary provides decision rules for resolving data problems associated with particular variables. It is meant to be a quick-reference guide of rules that can be used with any software platform. These

decision rules are then implemented in the step-by-step instructions the tasks provide.

SDP has also created a detailed **SDP Stata Glossary**, available as a separate document, that covers the Stata commands used throughout the toolkit. Commands are listed alphabetically and by subject.

As you go through a task, be sure to consult the data snippets in the left hand column of the page to get a visual sense for the changes occurring at each step.

Task Structure

The tasks follow a logical sequence from **1** to **7**. Each task comes with its own raw input file that results in a cleaned output file that matches or extends the file **Identify**. We also provide all cleaned output files so you can check your answers after completing each task. If you have followed the task instructions correctly, you should arrive at the same cleaned output file.

In each task, you will also find:

- **Purpose**: Clarifies the importance of the task.
- **How to Start**: Identifies the input file for the task.
- **Data Description**: Describes data elements for the task.
- **Instructions**: Provides instructions to tranform data. These instructions include:
  - Stata Code to help you execute the instructions through code.
  - Data Snapshots to help you visualize changes to the data at each step.

Summary

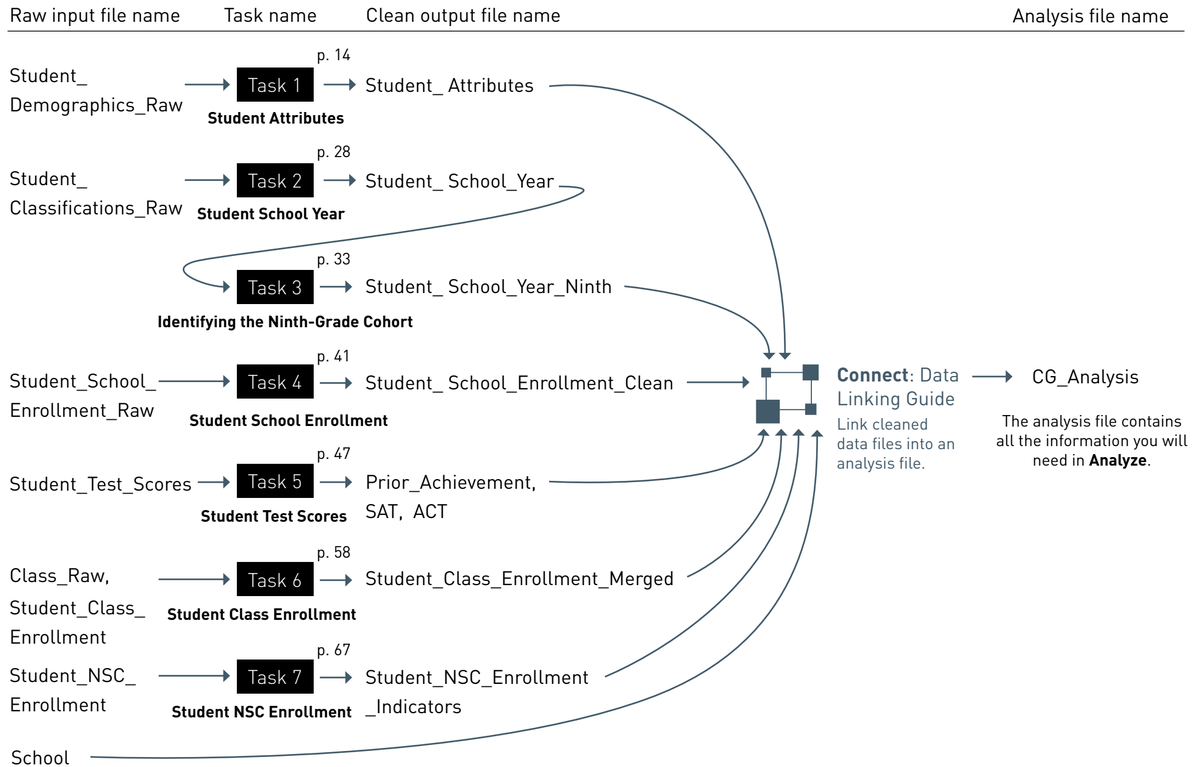
Through the tasks, you will learn effective practices for: data transformation, variable construction, and implementation of key decision rules.

The **Task Map** on the next page summarizes the inputs and outputs of each task and how the outputs are used in **Connect** to produce an analysis file. The Task Map also serves as a Table of Contents.

If you need additional guidance, the friendly research team at SDP is available to help: **sdp@gse.harvard.edu**.

Task Map

This map summarizes the inputs and outputs for each task and how the outputs are used in **Connect** to produce the college-going analysis and college-going analysis on-track file. The map also serves as a linked Table of Contents.



DECISION RULES GLOSSARY

This decision rules glossary provides a list of common data problems and SDP's decision rules for addressing them. It does not explain how to implement the rules step-by-step as the tasks provide this. The glossary can be a quick reference guide for recalling the decision rules and may be particularly useful for users of different analytic software.

Student_Attributes

Demographic, cohort, and graduation data for students.

Identifies unique observation: **sid**

Data Element	Possible Scenario	SDP Decision Rule	Reference in Data Building Tasks
sid			
male	Some students with conflicting genders may be observed.	If more than one gender observed, report the modal gender. If multiple modes, report the most recent gender recorded.	**Task 1**: Student Attributes
race_ethnicity	Some students with more than one race_ethnicity may be observed. For the purposes of analysis, SDP considers race_ethnicity time invariant.	If more than one category selected serially in time, report the modal race/ethnicity. If multiple modes are observed, report the most recent race/ethnicity recorded	**Task 1**: Student Attributes
race	Some students with more than one race may be observed. For the purposes of analysis, SDP considers race time invariant.	If more than one category selected serially in time, report the modal race. If multiple modes are observed, report the most recent race recorded.	
ethnicity	Some students with more than one ethnicity may be observed. For the purposes of analysis, SDP considers ethnicity time invariant.	If more than one ethnicity observed, report the modal ethnicity. If multiple modes are observed, report the most recent ethnicity recorded.	
birth_date	Some students with more than one birth date may be observed. For the purposes of analysis, SDP considers birth_date time invariant.	If more than one birth date observed, report the modal birth date. If multiple modes, report the most recent birth date recorded. When evaluating modal birth date exclude birth dates that fall outside +/- four years of the expected birth date given grade level and school year.	
first_9th_school_year_reported			
hs_diploma			**Task 1**: Student Attributes
hs_diploma_date	Some students with more than one diploma date may observed. For the purposes of analysis, SDP considers hs_diploma_date time invariant.	If more than one diploma date is observed, report the first diploma date.	**Task 1**: Student Attributes
hs_diploma_type	Some students with more than one diploma type may be observed. For the purposes of the analysis, SDP considers hs_diploma_type to be time invariant.	If more than one type is observed, report the type associated with the first diploma date. If multiple diploma types are observed for the first diploma date, report the modal value. If there is no mode, report the most competitive diploma type. If there is a diploma date but no diploma type, rpeort the diploma type as "Unknown."	**Task 1**: Student Attributes
zip_code			

Student_School_Year

Yearly classification and attendance data for students.

Identifies unique observation: **sid** + **school_year**

Data Element	Possible Scenario	SDP Decision Rule	Reference in Data Building Tasks
sid			
school_year			
grade_level	Some students with more than one grade level may be observed during a given school year.	If more than one grade level observed during a school year, report the highest grade level recorded.	**Task 2**: Student School Year
frpl	Some students with more than one free or reduced price lunch status may be observed during a given school year.	If more than one status is observed during the school year, report the highest non-null value.	**Task 2**: Student School Year
iep	Some students with more than one individualized education plan status may be observed during a given school year.	If both "no IEP"and "has IEP" are observed during the same school year, report "has IEP".	**Task 2**: Student School Year
iep_classification	Some students with more than one individualized education plan classification may be observed during a given school year.	If more than one classification is observed during the school year, report the last non-null classification reported.	
ell	Some students with more than one English language learner status may be observed during a given school year.	If both "not ell"and "ell" are observed in during the same school year, report "ell".	**Task 2**: Student School Year
ell_classification	Some students with more than one individualized education plan classification may be observed during a given school year.	If more than one classification is observed during the school year, report the last non-null classification reported.	
gifted	Some students with more than one gifted status may be observed during a given school year.	If both "not enrolled"and "enrolled" are observed during the same school year, report "enrolled".	**Task 2**: Student School Year
gifted_classification	Some students with more than one giftted classification may be observed during a given school year.	If more than one classification is observed during the school year, report the last non-null classification reported.	
total_ days_enrolled		If not reported, value may be calculated by number of school days between enrollment_date and withdrawal_date, or days_present + days_absent.	**Task 2**: Student School Year
total_days_absent		Cannot exceed the number of days enrolled.	**Task 2**: Student School Year
days_suspended_ out_of_school		Cannot exceed the number of days enrolled.	**Task 2**: Student School Year
days_suspended_in_ school		Cannot exceed the number of days enrolled.	**Task 2**: Student School Year

Student_School_Enrollment

School enrollment/withdrawal data for students.

Identifies unique observation: **sid** + **school_year** + **school_code + enrollment_date**

Data Element	Possible Scenario	SDP Decision Rule	Reference in Data Building Tasks
sid			
school_year			
school_code			
enrollment_date	Many different scenarios may be encountered: - Some enrollment spells may overlap at the same school. - Some enrollment spells may have a missing enrollment_date and withdrawal_date. - Some enrollment spells may have the same enrollment_date and withdrawal_date. - Some enrollment spells may begin after the withdrawal_date. - Some enrollment spells may begin after the end of the current school_year or before the beginning of the current school_year.	If enrollment spells overlap at the same school, consolidate enrollment observations into one enrollment period, using the earliest enrollment date and last withdrawal date and their corresponding codes and descriptions. Drop any other abnormal enrollment observations.	**Task 4**: Student School Enrollment
enrollment_code			
enrollment_code_ desc			
withdrawal_date			
withdrawal_code			
withdrawal_code_ desc			
days_enrolled		Update days_enrolled after you have consolidated overlapping enrollment observations.	**Task 4**: Student School Enrollment
days_present			

Student_Test_Scores

Standardized test data for students (state standardized tests, advanced placement, SAT, ACT, etc). Every attempt at a test by a student should be recorded.

Identifies unique observation: **sid** + **test_code** + **test_date**

Data Element	Possible Scenario	SDP Decision Rule	Reference in Data Building Tasks
sid			
test_code			
test_date	Students who re-take tests in the same year or are retained in a grade and re-take the same test in different years may have multiple observations for a single test code.	Take the earliest test in both cases.	**Task 5**: Student Test Scores
test_code_desc			
test_type			**Task 5**: Student Test Scores
grade_level			**Task 5**: Student Test Scores
test_subject			**Task 5**: Student Test Scores
test_version			
language_version			**Task 5**: Student Test Scores
raw_score	The raw score or scaled score may be missing or take on values outside the accepted range. Additionally, a test observation may have a missing raw test score but not a missing scaled, and vice versa.	Set scores outside the accepted range to missing. For instance, in the synthetic data, a test score of 0 is considered outside the accepted range and is set to missing. Based on knowledge of your agency, decide if all test observations should have both raw and scaled scores. One option would be to drop test observations missing either of the test scores. Here we only drop test observations that are missing both test scores.	**Task 5**: Student Test Scores
scaled_score			**Task 5**: Student Test Scores
performance_level			**Task 5**: Student Test Scores
standardized_score	The system may not provide a standardized score (i.e., mean zero, s.d. one).	Generate a standardized test score by subject, test_type, and school_year.	

Class

Class level scheduling data.

Identifies unique observation: **cid**

Data Element	Possible Scenario	SDP Decision Rule	Reference in Data Building Tasks
cid			**Task 6:** Student Class Enrollment
school_year			**Task 6:** Student Class Enrollment
school_code			**Task 6:** Student Class Enrollment
course_code			**Task 6:** Student Class Enrollment
course_code_desc	Agencies often have a very large number of course names. In some cases other criteria (e.g. department the course listed in, or length of course) is needed to identify a course.	Use agency rules to flag math and English courses.	**Task 6:** Student Class Enrollment
section_code			
period_bell			
room_number			
tid			
semester_term_year			**Task 6:** Student Class Enrollment
graduation_requirement			
credits_possible			**Task 6:** Student Class Enrollment
instructional_level			
subject			**Task 6:** Student Class Enrollment

Student_Class_Enrollment

Class enrollment, grades, and credits earned data for students.

Identifies unique observation: **sid** + **cid** + **enrollment_date**

Data Element	Possible Scenario	SDP Decision Rule	Reference in Data Building Tasks
sid			**Task 6:** Student Class Enrollment
cid			**Task 6:** Student Class Enrollment
class_enrollment_ date	Some class enrollment spells may overlap: students may have more than one class_enrollment_date and/or more than one class_withdrawal_date for the same cid, school_year and marking_period.	For overlapping class enrollment spells, report the earliest enrollment date and latest withdrawal date	**Task 6:** Student Class Enrollment
class_withdrawal_ date	Some class enrollment spells may overlap: students may have more than one class_enrollment_date and/or more than one class_withdrawal_date for the same cid, school_year and marking_period.	For overlapping class enrollment spells, report the earliest enrollment date and latest withdrawal date	**Task 6:** Student Class Enrollment
credits_earned			**Task 6:** Student Class Enrollment
final_grade_mark			**Task 6:** Student Class Enrollment

Student_NSC_Enrollment

National Student Clearinghouse Student Tracker student-level data that provides information on postsecondary outcomes.

Identifies unique observation: **sid** + **college_code_branch** + **enrollment_begin** + **enrollment_end**

Data Element	Possible Scenario	SDP Decision Rule	Reference in Data Building Tasks
sid			**Task 7:** Student NSC Enrollment
college_code_branch			**Task 7:** Student NSC Enrollment
enrollment_begin		Identify the student's first college by the earliest enrollment date.	**Task 7:** Student NSC Enrollment
enrollment_end			**Task 7:** Student NSC Enrollment
record_found_yn			**Task 7:** Student NSC Enrollment
high_school_code			
high_school_grad_dt			
college_name			**Task 7:** Student NSC Enrollment
college_state		Create an indicator for in-state and one for out of state colleges	**Task 7:** Student NSC Enrollment
year4year		Create two indicators to mark 4-year colleges and 2-year colleges. Combine "2-year" and "Less Than 2 Years".	**Task 7:** Student NSC Enrollment
public_private		Create two indicators for the two types.	**Task 7:** Student NSC Enrollment
enrollment_status		Sometimes students enroll in more than one college at the same time. When identifying first college, report the one with the highest enrollment status (F, H, L, in order of importance) and then by longest duration.	**Task 7:** Student NSC Enrollment
graduated			**Task 7:** Student NSC Enrollment
graduation_date			**Task 7:** Student NSC Enrollment
college_sequence			**Task 7:** Student NSC Enrollment
degree_title			**Task 7:** Student NSC Enrollment
major			**Task 7:** Student NSC Enrollment

Task 1 STUDENT ATTRIBUTES

PURPOSE

In **Task 1**: Student Attributes, you will take the Student_Demographics_Raw file and generate the clean Student_Attributes file that matches the specification in **Identify** with one observation per student.

The core of this task:

1. Create consistent gender indicators for students across years.

2. Create consistent race/ethnicity values for students across years.

3. Create consistent values for high school diploma indicators.

HOW TO START

To begin, open the Student_Demographics_Raw file in Stata. If you do not have Stata, you can follow the steps of the task by looking at the instructions and data snippets we have provided.

If this is your first time attempting **Task 1**, start with the provided raw input file. This file teaches you SDP's cleaning methodology and allows you to check answers from a common dataset.

Task 1 STUDENT ATTRIBUTES

DATA DESCRIPTION FOR RAW FILE

The clean Student_Attributes file includes **sid, male, race_ethnicity, first_9th_school_year_reported, hs_diploma, hs_diploma_date** and **hs_diploma_type**. Later analyses do not currently make use of birth dates and zip codes, and these variables are thus excluded. This file contains the combined race_ethnicity variable rather than separate variables for race and ethnicity.

The raw input file, Student_Demographics_Raw, varies from the clean Student_Attributes file in a number of ways. In Student_Demographics_Raw, **race_ethnicity** is coded as a string rather than numeric and does not distinguish between the designations multiple, "M", and other, "O". Student_Demographics_Raw is also a time-variant data set including **school_year** so the data is unique by **sid** and **school_year.** Student_Attributes, however, is unique by **sid** alone. The aim of this task will be to match Student_Attributes to be unique by **sid** only.

Field Name	Values or Data Type	Definition
sid	numeric	Student identifier unique to each student. This identification number is typically assigned to a student upon enrollment in your agency. State agencies may have different identification numbers than district agencies for the same student.
school_year	spring calendar year	Academic school year from fall to spring, denoted here as the spring calendar year.
male	0 = female 1 = male	Student gender.
race_ethnicity	"B" = African American "A" = Asian American "H" = Hispanic "NA" = American Indian "WH" = White, not Hispanic "M/O"= Multiple/Other	If the system allows the indication of multiple categories simultaneously (e.g., African American and white) report "multiple.", unless one of the categories is "Hispanic", in which case Hispanic will be reported. (This rule follows federal standards for reporting race and ethnicity). **In this case, the variable is coded as a string. Part of this task involves recoding the variable as numeric to bring it in line with Identify.**
first_9th_school_year_reported	Spring calendar year	Academic school year from fall to spring, denoted here as the spring calendar year. For each student, the first school year in which they were 9th graders will be reported.
hs_diploma	0 = no high school diploma 1 = has high school diploma	Indicates whether the student earned a high school diploma
hs_diploma_date	date format (yyyy-mm-dd)	First recorded diploma type
hs_diploma_type	Local values	First recorded diploma type

Uniqueness: Some agencies may record **race_ethnicity** and/or **gender** each school year. Alternatively, students may have multiple records for having attended ninth grade or multiple diploma dates and/or types. To fix this issue, you will create a Student_Attributes research file unique by **sid** alone starting from a Student_Demographics_Raw file that is unique by **sid** and **school_year**. Once the file is unique by **sid** as shown in **Identify**, it is ready for **Connect**.

Task 1 STUDENT ATTRIBUTES

/*** Step 0: Load the Student_Demographics_Raw data file ***/

sid	school_year	male	race_ ethnicity	first_9th_ school_year _reported	hs_ diplo- ma	hs_ diploma_ date	hs_ diploma_type
2	2006	0	W	2005	1	14-May-08	Standard Diploma
2	2007	0	B	2005	1	14-May-08	Standard Diploma
8552	2005	1	W	2005	0	.	
8552	2006	0	A	2005	0	.	
8552	2006	1	W	2005	0	.	
8552	2007	1	W	2005	0	.	
8552	2009	1	W	2005	0	.	

```
use "${raw}\Student_Demographics_Raw.dta", clear
```

// Drop the first_9th_school_year_reported variable. You will create a first_9th_grade_observed variable in Task 3 that also imputes the first 9th grade for transfer-ins.

sid	school_year	male	race_ ethnicity	first_9th_ school_year _reported	hs_ diplo- ma	hs_ diploma_ date	hs_ diploma_type
2	2006	0	W	2005	1	14-May-08	Standard Diploma
2	2007	0	B	2005	1	14-May-08	Standard Diploma
8552	2005	1	W	2005	0	.	
8552	2006	0	A	2005	0	.	
8552	2006	1	W	2005	0	.	
8552	2007	1	W	2005	0	.	
8552	2009	1	W	2005	0	.	

```
drop first_9th_school_year_reported
```


Task 1 STUDENT ATTRIBUTES

/*** Step 1: Create one consistent value for gender for each student across years. ***/

// 1. Create a variable that shows how many unique values male assumes for each student. Name this variable nvals_male. Tabulate the variable and browse the relevant data.

Focus on the sid, school_year, and male variables. The other variables remain in the dataset.

sid	school_year	male	nvals_ male
8552	2005	1	2
8552	2006	0	2
8552	2006	1	2
8552	2007	1	2
8552	2009	1	2
12506	2004	1	2
12506	2005	0	2

```
bys sid: egen nvals_male = nvals(male)
tab nvals_male
```

```
 nvals_male |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |     87,517       99.98       99.98
          2 |         17        0.02      100.00
------------+-----------------------------------
      Total |     87,534      100.00
```

```
br if nvals_male > 1
```

// 2. Identify the modal gender. If multiple modes exist for a student, report the most recent gender recorded.

sid	school_year	male	nvals_ male	male_ mode
8552	2005	1	2	1
8552	2006	0	2	1
8552	2006	1	2	1
8552	2007	1	2	1
8552	2009	1	2	1
12506	2004	1	2	.
12506	2005	0	2	.

```
// Define the modal gender. For students who have a mode, replace male with the modal value (male_mode will be missing if there is no single mode).

bys sid: egen male_mode = mode(male)
```

Task 1 STUDENT ATTRIBUTES

sid	school_year	male	nvals_ male	male_ mode
8552	2005	1	2	1
8552	2006	1	2	1
8552	2006	1	2	1
8552	2007	1	2	1
8552	2009	1	2	1
12506	2004	1	2	.
12506	2005	0	2	.

Let's look just at sid==12506 which doesn't have a single mode.

sid	school_year	male	nvals_ male	male_ mode
12506	2005	0	2	.
12506	2004	1	2	.

sid	school_year	male	nvals_ male	male_ mode	temp_male_ last	male_ last
12506	2005	0	2	.	0	0
12506	2004	1	2	.	.	0

sid	school_year	male	nvals_ male	male_ mode	temp_male_ last	male_ last
12506	2005	0	2	.	0	0
12506	2004	0	2	.	.	0

```
replace male = male_mode if !mi(male_mode)
```

```
// If multiple modes exist for a student, report the most
recent gender recorded

gsort sid -school_year
```

```
bys sid: gen temp_male_last = male if _n==1
bys sid: egen male_last = max(temp_male_last)
```

```
replace male = male_last if mi(male_mode)
```


Task 1 STUDENT ATTRIBUTES

sid	school_year	male
8552	2005	1
8552	2006	1
8552	2006	1
8552	2007	1
8552	2009	1
12506	2005	0
12506	2004	0

```
// Drop temporary variables

drop nvals_male male_mode temp_male_last male_last
```

/*** Step 2: Create one consistent value for race_ethnicity for each student across years. ***/

// 1. Recode the raw race_ethnicity variable as a numeric variable and label it. Replace the string race_ethnicity variable with the numeric one.

Focus on the sid, school_year, and race_ethnicity variables. The other variables remain in the dataset.

sid	school_year	race_ num
8552	2005	W
8552	2006	A
8552	2006	W
8552	2007	W
8552	2009	W

```
generate race_num=.
replace race_num = 1 if race_ethnicity==”B”
replace race_num = 2 if race_ethnicity ==”A”
replace race_num = 3 if race_ethnicity ==”H”
replace race_num = 4 if race_ethnicity ==”NA”
replace race_num = 5 if race_ethnicity ==”W”
replace race_num = 6 if race_ethnicity ==”M/O”

order race_num, after(race_ethnicity)

label define race 1 “Black” 2 “Asian” 3 “Hispanic” 4 “Native
American” 5 “White” 6 “Multiple/Other”
label val race_num race
```

Task 1 STUDENT ATTRIBUTES

sid	school_year	race_ ethnicity
8552	2005	W
8552	2006	A
8552	2006	W
8552	2007	W
8552	2009	W

```
tab race_num, mi

       race_num |      Freq.     Percent        Cum.
----------------+-----------------------------------
          Black |     25,321       28.93       28.93
          Asian |      7,303        8.34       37.27
       Hispanic |     30,444       34.78       72.05
Native American |      1,129        1.29       73.34
          White |     20,528       23.45       96.79
 Multiple/Other |      2,809        3.21      100.00
----------------+-----------------------------------
          Total |     87,534      100.00

drop race_ethnicity
rename race_num race_ethnicity
```

// 2. Create a variable that shows how many unique values race_ethnicity assumes for each student. Name this variable nvals_race. Tabulate the variable.

sid	school_year	race_ ethnicity	nvals_ race
8552	2005	W	2
8552	2006	A	2
8552	2006	W	2
8552	2007	W	2
8552	2009	W	2

```
bys sid: egen nvals_race = nvals(race_ethnicity)
tab nvals_race

 nvals_race |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |     87,176       99.59       99.59
          2 |        328        0.37       99.97
          3 |         30        0.03      100.00
------------+-----------------------------------
      Total |     87,534      100.00
```

// 3. Create a variable that shows how many unique values race_ethnicity assumes for each student and school_year. Name this variable nvals_race_yr. Tabulate the variable and browse the relevant data.

sid	school_year	race_ ethnicity	nvals_ race	nvals_ race_yr
8552	2005	W	2	1
8552	2006	A	2	2
8552	2006	W	2	2
8552	2007	W	2	1
8552	2009	W	2	1

```
bys sid school_year: egen nvals_race_yr = nvals(race_ethnic-
ity)
tab nvals_race_yr
br if nvals_race_yr > 1
```


Task 1 STUDENT ATTRIBUTES

// 4. If more than one race is reported in the same school_year, report students as multiracial, unless one of their reported race_ethnicity values is Hispanic. Report the student as Hispanic in that case.

sid	school_year	race_ ethnicity	nvals_ race	nvals_ race_yr	temp_ ishispanic	ishispanic
8552	2005	W	2	1	.	.
8552	2006	A	2	2	.	.
8552	2006	W	2	2	.	.
8552	2007	W	2	1	.	.
8552	2009	W	2	1	.	.

```
gen temp_ishispanic = .
replace temp_ishispanic = 1 if race_ethnicity == 3 & nvals_
race_yr > 1
bys sid school_year: egen ishispanic = max(temp_ishispanic)
```

sid	school_year	race_ ethnicity	nvals_ race	nvals_ race_yr	temp_ ishispanic	ishispanic
8552	2005	W	2	1	.	.
8552	2006	M/O	2	2	.	.
8552	2006	M/O	2	2	.	.
8552	2007	W	2	1	.	.
8552	2009	W	2	1	.	.

```
replace race_ethnicity = 3 if nvals_race_yr > 1 & ishispanic
== 1
replace race_ethnicity = 6 if nvals_race_yr > 1 & ishispanic
!= 1
```

sid	school_year	race_ ethnicity	nvals_ race
8552	2005	W	2
8552	2006	M/O	2
8552	2007	W	2
8552	2009	W	2

```
// Drop the temporary variables we created
drop temp_ishispanic ishispanic

// Drop the duplicates resulting from fixing student with
different race_ethnicity in a school_year
duplicates drop if nvals_race_yr > 1
drop nvals_race_yr
```

// 5. Report the modal race. If multiple modes exist for a student, report the most recent race recorded.

sid	school_year	race_ ethnicity	nvals_ race	race_ mode
8552	2005	W	2	W
8552	2006	M/O	2	W
8552	2007	W	2	W
8552	2009	W	2	W

```
// Identify the modal race. For students who have a mode,
replace race with the modal value (race_mode will be missing
if there is no single mode).

bys sid: egen race_mode = mode(race_ethnicity)
```

sid	school_year	race_ ethnicity	nvals_ race	race_ mode
8552	2005	W	2	W
8552	2006	W	2	W
8552	2007	W	2	W
8552	2009	W	2	W

```
replace race_ethnicity = race_mode if !mi(race_mode)
```

Now consider sid==2.

sid	school_year	race_ ethnicity	nvals_ race	race_ mode
2	2006	W	2	.
2	2007	B	2	.

sid	school_year	race_ ethnicity	nvals_ race	race_ mode	temp_race_last	race_ last
2	2007	B	2	.	B	B
2	2006	W	2	.	.	B

sid	school_year	race_ ethnicity	nvals_ race	race_ mode	temp_race_last	race_ last
2	2007	W	2	.	B	B
2	2006	W	2	.	.	B

sid	school_year	race_ ethnicity
8552	2005	W
8552	2006	W
8552	2007	W
8552	2009	W
2	2007	B
2	2006	B

```
// If multiple modes exist for a student, report the most
recent race recorded

gsort sid -school_year
bys sid: gen temp_race_last = race_ethnicity if _n==1
bys sid: egen race_last = max(temp_race_last)

replace race_ethnicity = race_last if mi(race_mode)
```

```
// Drop the duplicates resulting from fixing student with
different race_ethnicity in a school_year

duplicates drop if nvals_race_yr > 1
drop nvals_race_yr

// Drop temporary variables

drop nvals_race race_mode temp_race_last race_last

// Examine the distribution of race_ethnicity in your agen-
cy.

tab race_ethnicity, mi
```

race_ethnicity	Freq.	Percent	Cum.
Black	25,323	28.93	28.93
Asian	7,262	8.30	37.23
Hispanic	30,443	34.78	72.01
Native American	1,132	1.29	73.30
White	20,553	23.48	96.78
Multiple/Other	2,818	3.22	100.00
Total	87,531	100.00	

Task 1 STUDENT ATTRIBUTES

/*** Step 3: Create consistent values for high school diploma variables. ***/

// 1. Recode the hs_diploma_type variable as a numeric variable and label it. Replace the string hs_diploma_type variable with the numeric one. Use lower numbers for more competitive diploma types.

Focus on the sid, school_year, and hs_diploma variables. The other variables remain in the dataset.

sid	school_year	hs_diploma	hs_diploma_date	hs_diploma_type	dipl_num
16	2007	1	14-May-08	Standard Diploma	2
16	2008	1	14-May-08	College Prep Diploma	2

```
generate dipl_num =.
replace dipl_num = 1 if hs_diploma_type==”College Prep Diploma”
replace dipl_num = 2 if hs_diploma_type==”Standard Diploma”
replace dipl_num = 3 if hs_diploma_type==”Alternative Diploma”

order dipl_num, after(hs_diploma_type)
```

sid	school_year	hs_diploma	hs_diploma_date	dipl_num
16	2006	1	14-May-08	2
16	2007	1	14-May-08	1

```
label define dipl 1 “College Prep Diploma” 2 “Standard Diploma” 3 “Alternative Diploma” 4 “Unknown”
label val dipl_num dipl
```

sid	school_year	hs_diploma	hs_diploma_date	diploma_type
16	2006	1	14-May-08	2
16	2007	1	14-May-08	1

```
drop hs_diploma_type
rename dipl_num hs_diploma_type
```

// 2. Identify the first diploma date reported

sid	school_year	hs_diploma	hs_diploma_date	diploma_type	temp_earliest_dipl_date	earliest_dipl_date
16	2006	1	14-May-08	2	14-May-08	14-May-08
16	2007	1	14-May-08	1	.	14-May-08

```
sort sid hs_diploma_date
bys sid: gen temp_earliest_dipl_date = hs_diploma_date if _n==1
bys sid: egen earliest_dipl_date = max(temp_earliest_dipl_date)
```

// 3. Identify the diploma type associated with the first diploma date

Task 1 STUDENT ATTRIBUTES

sid	school_year	hs_diploma	hs_diploma_date	diploma_type	temp_earliest_dipl_date	earliest_dipl_date	earliest_dipl_type
16	2006	1	14-May-08	2	14-May-08	14-May-08	2
16	2007	1	14-May-08	1	.	14-May-08	.

```
gen earliest_dipl_type = hs_diploma_type if hs_diploma_date == earliest_dipl_date
```

// 4. Create a variable that shows the number of unique diploma types recorded for the first diploma date.

sid	school_year	...	diploma_type	temp_earliest_dipl_date	earliest_dipl_date	earliest_dipl_type	nvals_dipl_type
16	2006	...	2	14-May-08	14-May-08	2	1
16	2007	...	1	.	14-May-08	.	.

```
bys sid: egen nvals_dipl_type = nvals(earliest_dipl_type)
tab nvals_dipl_type
```

The hs_diploma and hs_diploma_date are still in the dataset: they've just been removed to make space for other variables.

// 5. Identify the modal diploma type. If multiple modes exist for a student, report the diploma type in the earliest school year for the first diploma date.

sid	school_year	...	diploma_type	temp_earliest_dipl_date	earliest_dipl_date	earliest_dipl_type	nvals_dipl_type	dipl_type_mode
16	2006	...	2	14-May-08	14-May-08	2	1	.
16	2007	...	1	.	14-May-08	.	.	.

```
bys sid: egen dipl_type_mode = mode(earliest_dipl_type)
replace hs_diploma_type = dipl_type_mode if !mi(dipl_type_mode)
```

sid	school_year	...	diploma_type	temp_earliest_dipl_date	earliest_dipl_date	earliest_dipl_type	nvals_dipl_type	dipl_type_mode	temp_dpl	earliest_type_by_earliest_date
16	2006	...	2	14-May-08	14-May-08	2	1	.	2	2
16	2007	...	1	.	14-May-08	.	.	.	.	2

```
sort sid earliest_dipl_date school_year
bys sid: gen temp_dpl = earliest_dipl_type if _n==1 & school_year[_n] != school_year[_n+1]
bys sid: egen earliest_type_by_earliest_date = mode(temp_dpl)
```

sid	school_year	...	diploma_type	temp_earliest_dipl_date	earliest_dipl_date	earliest_dipl_type	nvals_dipl_type	dipl_type_mode	temp_dpl	earliest_type_by_earliest_date
16	2006	...	2	14-May-08	14-May-08	2	1	.	2	2
16	2007	...	2	.	14-May-08	.	.	.	.	2

```
replace hs_diploma_type = earliest_type_by_earliest_date if mi(dipl_type_mode) & !mi(earliest_type_by_earliest_date)
```

// 6. If multiple diploma types were recorded for the same school year and first diploma date, report the most competitive diploma type.

sid	school_year	...	diploma_type	temp_earliest_dipl_date	earliest_dipl_date	earliest_dipl_type	nvals_dipl_type	dipl_type_mode	temp_dpl	earliest_type_by_earliest_date	isdiffdipl	temp_most_compet
20	2008	...	2	14-May-08	14-May-08	2	.	.	.	.	1	2
20	2008	...	1	.	14-May-08	.	.	.	.	.	1	2

```
gen isdiffdipl = 1 if !mi(earliest_dipl_date) & mi(dipl_
type_mode) & mi(earliest_type_by_earliest_date)
gen temp_most_compet = .
replace temp_most_compet = earliest_dipl_type if isdiffdipl
== 1
bys sid: egen most_compet = min(temp_most_compet)
```

sid	school_year	...	diploma_type	temp_earliest_dipl_date	earliest_dipl_date	earliest_dipl_type	nvals_dipl_type	dipl_type_mode	temp_dpl	earliest_type_by_earliest_date	isdiffdipl	temp_most_compet
20	2008	...	2	14-May-08	14-May-08	2	.	.	.	.	1	2
20	2008	...	2	.	14-May-08	.	.	.	.	.	1	2

```
replace hs_diploma_type =  most_compet if isdiffdipl == 1
```

// 7. If there are any missing diploma types, mark these as an unknown diploma type.

```
replace hs_diploma_type = 4 if mi(hs_diploma_type) & !mi(hs_diploma_date)
```

// 8. Finally, replace hs_diploma_date with the first hs_diploma_date

```
replace hs_diploma_date = earliest_dipl_date
```

// 9. Make sure that diploma is set to 1 if there is a diploma date reported

```
replace hs_diploma = 1 if !mi(hs_diploma_date)
```

// 10. Drop all temporary variables we created

```
drop temp_earliest_dipl_date- most_compet
```


/**** Step 4: Drop any unneeded variables, drop duplicates, check the data, and save the file ****/

// 1. Drop school_year as you no longer need it. Also drop birth_date since it is not used in later analyses.

```
drop school_year birth_date
```

// 2. Drop duplicate values

```
duplicates drop
```

// 3. Check that the file is unique by sid

```
isid sid
```

// 4. Save the current file as Student_Attributes.dta

```
save "${clean}\Student_Attributes.dta", replace
```

PURPOSE

In **Task 2:** Student School Year, you will take the Student_Classifications_Raw file and generate a clean Student_School_Year output file that matches the specification in **Identify** with one observation per student and school year. To do so, you will first ensure only one grade level is assigned per student per school year. Then, you will process the free or reduced price lunch (FRPL) variable (a proxy for students' poverty status), individualized education program (IEP) variable, English language learner (ELL) variable, and gifted variable. You will also examine the total days enrolled, days absent, and days suspended variables.

The core of this task:

1. Resolve instances when students have more than one grade level in a school year
2. Keep the highest value of FRPL reported by student by school year
3. If a student has both "has IEP" and "no IEP" reported in a school year, keep "has IEP"
4. If a student has both "has ELL" and "no ELL" reported in a school year, keep "has ELL"
5. If a student is observed as both gifted eligible and not eligible, report eligible
6. Explore days_enrolled, days_absent and days_suspended
7. Drop duplicate observations to make the file unique by student and school year

After this, you will have a data set unique by student and school year that allows you to assign students to the appropriate ninth grade cohort in **Task 3**.

HOW TO START

To begin, open the Student_Classifications_Raw file in Stata. If you do not have Stata, you can follow the steps of the task by looking at the instructions and data snippets we have provided.

If this is your first time attempting **Task 2**, start with the provided raw input file. This file teaches you SDP's cleaning methodology and allows you to check answers from a common dataset.

DATA DESCRIPTION FOR RAW FILE

The raw input file, Student_Classifications_Raw, varies from the clean Student_School_Year file in that it is not unique by **sid** and **school_year** as shown in **Identify**. For the same **sid** and **school_year**, a student may have more than one grade_level, frpl, iep, ell, or gifted status reported. The aim of this task will be to match Student_School_Year in **Identify** in its structure and uniqueness so it is unique by **sid** and **school_year**.

Field Name	Values or Data Type	Definition
sid	numeric	Student identifier unique to each student. This identification number is typically assigned to a student upon enrollment in your agency. State agencies may have different identification numbers than district agencies for the same student.
school_year	spring calendar year	Academic school year from fall to spring, denoted here as the spring calendar year.
grade_level	-9 = ungraded -1 = any pre-kindergarten 0 = kindergarten 1-12 = grades 1–12 13+ = additional grade levels (i.e. vocational training, special education past year 12)	Grade level of student.
frpl	"N" = not participating "R" = reduced lunch "F" = free lunch null = no status	Status in the free or reduced price lunch program. **In this case, the variable is coded as a string. Part of this task will involve recoding the variable as numeric to create an indicator for a student having ever been FRPL.**
iep	0 = no IEP 1 = has IEP	Indicator for students who have an individualized education plan (IEP).
ell	0 = not ell 1 = ell	Indicator for students who are classified as English Language Learners (ELL). Some systems refer to this category as Limited English Proficient (LEP) or English as a Second Language (ESL).
gifted	not enrolled in a gifted 0 = education program 1 = enrolled in a gifted education program	Indicator variable for students enrolled in gifted and talented education programs .
days_enrolled	number of days	Number of days over the school year a student was enrolled at a school.
days_absent	number of days	Number of days over the school year a student was marked absent. Cannot exceed the number of days enrolled.
days_suspended_out_of_school	number of days	Total number of days over the school year a student experienced out of school suspension.

Uniqueness: Some students transfer grades mid-year and therefore report two grade levels within a school year. Furthermore, as they transfer, other attributes might change. There may also be duplicates due to clerical errors. To address this issue, you will take the Student_Classifications_Raw file and make it unique by **sid** and **school_year**.

Task 2 STUDENT SCHOOL YEAR

/*** Step 0: Load the Student_Classifications_Raw data file ***/

sid	school_ year	grade_ level	frpl	iep	ell	gifted	total_ days_ enrolled	total_ days_ab- sent	days_ suspended_ out_of_ school
3	2007	8	F	0	0	0	228	N	0
3	2007	9	F	0	0	0	228	N	0

```
use "${raw}\Student_Classifications_Raw.dta", clear
```

/*** Step 1: Create one consistent grade level for each student within the same year. ***/

// 1. Keep the highest grade_level when a student has multiple grade levels within the same year.

Focus on the sid, school_year, and grade_level variables. The other variables remain in the dataset.

sid	school_ year	grade_ level	nvals_ grade	max_grade_ level
3	2007	8	2	9
3	2007	9	2	9

sid	school_ year	grade_ level	nvals_ grade	max_grade_ level
3	2007	9	2	9
3	2007	9	2	9

sid	school_ year	grade_ level
3	2007	9
3	2007	9

```
// Check if there are any instances of multiple grade levels
per sid per school_year

bys sid school_year: egen nvals_grade = nvals(grade_level)
tab nvals_grade

// Keep the highest value per school year

bys sid school_year: egen max_grade_level = max(grade_level)

replace grade_level = max_grade_level


// drop temporary variables

drop nvals_grade max_grade_level
```


Task 2 STUDENT SCHOOL YEAR

/*** Step 2: Create one consistent FRPL value for each student within the same year. ***/

// 1. Recode raw frpl variable with string type to numeric type

sid	school_ year	grade_ level	frpl	frpl_ num
80	2005	9	N	0
80	2005	9	R	1

sid	school_ year	grade_ level	frpl
80	2005	9	0
80	2005	9	1

```
gen frpl_num = .
replace frpl_num = 0 if frpl=="N"
replace frpl_num = 1 if frpl=="R"
replace frpl_num = 2 if frpl=="F"

order frpl_num, after(frpl)

// Drop the old string variable and rename the numeric vari-
able as frpl

drop frpl
rename frpl_num frpl
```

// 2. Ensure that frpl is consistent by sid and school_year. In cases where multiple values exist, report the highest value.
Follow the same procedure as Step 1 for grade_level.

```
// Check if there are any cases where different values of frpl status are reported in a year

bys sid school_year: egen nvals_frpl = nvals(frpl)
tab nvals_frpl

// Report the highest value of frpl by year for each student, selecting free over reduced over not participating.

egen highest_frpl = max(frpl), by(sid school_year)
replace frpl = highest_frpl

// Label the values so they are easy to understand

label define frpl 0 "Not FRPL eligible" 1 "Reduced price lunch eligible" 2 "Free price lunch eligible"

label values frpl frpl

// Drop the temporary values we created

drop nvals_frpl highest_frpl
```

/*** Step 3: Create one consistent IEP value for each student within the same year. ***/
Follow the same procedure as Step 1 for grade_level.

```
// Report the highest value of iep by year for each student, selecting has iep over not iep.

egen highest_iep = max(iep), by(sid school_year)
replace iep = highest_iep

drop highest_iep
```

/*** Step 4: Create one consistent ELL value for each student within the same year. ***/
Follow the same procedure as Step 1 for grade_level.

```
// Report the highest value of ell by year for each student, selecting is ell over not ell.

egen highest_ell = max(ell), by(sid school_year)
replace ell = highest_ell

drop highest_ell
```

/*** Step 5: Create one consistent gifted value for each student within the same year. ***/
Follow the same procedure as Step 1 for grade_level.

```
// Report the highest value of gifted by year for each student, selecting is enrolled in gifted program over not enrolled.

egen highest_gifted = max(gifted), by(sid school_year)
replace gifted = highest_gifted

drop highest_gifted
```

/**** Step 6: Drop any unneeded variables, drop duplicates, and save the file ****/

```
// 1. Drop duplicate observations

duplicates drop

// 2. Make sure your file is now unique by student and school year

isid sid school_year

// 3. Save the current file as Student_School_Year.dta which you will need for Task 3.

save "${clean}\Student_School_Year.dta", replace
```


PURPOSE

In **Task 3**: Identifying the Ninth Grade Cohort, you will identify the school year students first appear in ninth grade using the clean Student_School_Year research file from **Task 2**. This essential step allows you to form student cohorts and examine longitudinal college-going outcomes.

The core of this task:

1. Flag the first school year a student enrolls in grades 9, 10, 11, or 12.

2. Identify the school year in which the student was first observed in 9th grade.

3. Impute the school year in which transfer students would have been in grade 9.

4. Replace the first_9th_school_year_observed with the correctly imputed values.

After completing this task, you will have a clean Student_School_Year file that identifies first-time ninth graders. This file is used both to assemble the analysis file in **Connect** and to complete **Task 4**.

HOW TO START

To begin, open the Student_School_Year file, just created in **Task 2**, in Stata. If you do not have Stata, you can follow the steps of the task by looking at the instructions and data snippets we have provided.

If this is your first time attempting **Task 3**, start with the cleaned output file from **Task 2**. This file teaches you SDP's cleaning methodology and allows you to check answers from a common dataset.

DATA DESCRIPTION

The input file in this case, Student_School_Year, also the output from Task 2, now follows the structure of Student_School_Year in **Identify** so it is unique by **sid** and **school_year**. The aim of this task will be to create a **first_9th_school_year_observed** variable using the variables in the file.

Field Name	Values or Data Type	Definition
sid	numeric	Student identifier unique to each student. This identification number is typically assigned to students upon enrollment in your agency. State agencies may have different identification numbers than district agencies for the same student.
school_year	spring calendar year	Academic school year from fall to spring, denoted here as the spring calendar year.
grade_level	1-12	Student grade level
frpl	0 = not participating 1 = reduced lunch 2 = free lunch null = no status	Status in the free or reduced price lunch program.
iep	0 = no IEP 1 = has IEP	Indicator for students who have an individualized education plan (IEP).
iep_classification	use local values	Local IEP or special education classification. Generally these classifications follow the standard special education classifications.
ell	0 = not ell 1 = ell	Indicator for students who are classified as English Language Learners (ELL). Some systems refer to this category as Limited English Proficient (LEP) or English as a Second Language (ESL).
ell_classification	use local values	Local classification of level of English language learner status.
gifted	0 = not enrolled in a gifted education program 1 = enrolled in a gifted education program	Indicator variable for students enrolled in gifted and talented education programs.
gifted_classification	use local values	Local classification, if any, for gifted eligible students.
total_days_enrolled	number of days	Total number of days over the school year a student was enrolled.
total_days_present	number of days	Total number of days over the school year a student was present. Cannot exceed the number of days enrolled.
total_days_absent	number of days	Total number of days over the school year a student was marked absent. Cannot exceed the number of days enrolled.
days_suspended_out_of_school	number of days	Total number of days over the school year a student experienced out of school suspension.

Uniqueness: This dataset was cleaned in **Task 2** and is now unique by **sid** and **school_year.**

Unique By:
sid + school_year

Student_School_Year
matching **Identify**, from **Task 2**

/*** Step 0: Load the Student_School_Year data file ***/

sid	school_year	grade_level
1	2004	9
1	2005	9
1	2006	10
1	2007	11

```
use "${clean}\Student_School_Year.dta", clear
```

/*** Step 1: Flag the first school year a student enrolls in grades 9, 10, 11, or 12. ***/

// Create four binary indicators to flag the first school year a student enrolls in grades 9, 10, 11, or 12.

sid	school_year	grade_level	first9_flag	observed_9
1	2004	9	1	1
1	2005	9	.	1
1	2006	10	.	1
1	2007	11	.	1

```
foreach grade in 9 10 11 12 {
      sort sid grade_level school_year

      bys sid grade_level: gen first`grade'_flag = _n ==1 if
grade_level == `grade'

      bys sid: egen observed_`grade' = max(first`grade'_flag)

      replace observed_`grade' = 0 if observed_`grade'== .
}
```

The same process occurs for grades 10, 11, and 12.

// Check how many students are identified as enrolled in grades 9, 10, 11, or 12

```
foreach grade in 9 10 11 12 {
      tab observed_`grade' if sid!=sid[_n-1],mi
}
```

```
observed_9 |      Freq.     Percent        Cum.
------------+-----------------------------------
         0 |      2,959       13.57       13.57
         1 |     18,844       86.43      100.00
------------+-----------------------------------
     Total |     21,803      100.00

observed_10 |      Freq.     Percent        Cum.
------------+-----------------------------------
         0 |      6,590       30.23       30.23
         1 |     15,213       69.77      100.00
------------+-----------------------------------
     Total |     21,803      100.00
```

```
observed_11 |      Freq.     Percent        Cum.
------------+-----------------------------------
         0 |     12,510       57.38       57.38
         1 |      9,293       42.62      100.00
------------+-----------------------------------
     Total |     21,803      100.00

observed_12 |      Freq.     Percent        Cum.
------------+-----------------------------------
         0 |     16,277       74.65       74.65
         1 |      5,526       25.35      100.00
------------+-----------------------------------
     Total |     21,803      100.00
```

/*** Step 2: Identify the school year in which the student was first observed in 9th grade. ***/

// Create a variable that lists the first school year a student is observed as enrolled in grade 9.

sid	school_year	grade_level	first9_flag	observed_9	temp_first9year	first_9th_school_year_observed
1	2004	9	1	1	2004	2004
1	2005	9	.	1	.	2004
1	2006	10	.	0	.	2004
1	2007	11	.	0	.	2004

```
bys sid: egen temp_first9year = min(school_year) if grade_
level == 9

bys sid: egen first_9th_school_year_observed = max(temp_first-
9year)

// Check the distribution of first_9th_school_year_observed
across years
sort sid
tab first_9th_school_year_observed if sid!=sid[_n-1]
```

```
 first_9th_s |
chool_year_ |
   observed |      Freq.     Percent        Cum.
------------+-----------------------------------
       2004 |          1        0.01        0.01
       2005 |      4,884       25.92       25.92
       2006 |      4,405       23.38       49.30
       2007 |      4,524       24.01       73.31
       2008 |      5,018       26.63       99.94
       2009 |         12        0.06      100.00
------------+-----------------------------------
      Total |     18,844      100.00
```


/*** Step 3: Impute the school year in which transfer students would have been in grade 9. ***/

// Impute first_9th_school_year_observed as school_year - 1, school_year - 2, or school_year - 3 for students first observed in 10th, 11th or 12th grade as transfer-ins

sid	school_year	grade_level	...	first_9th_school_year_observed	temp2_first9year	temp3_first9year
2	2006	10	...	.	2005	2005
2	2007	11	...	.		2005

The first`grade'_flag, observed_`grade', and temp_first`grade'year variables are still in the dataset: they've just been removed to make space for other variables.

sid	school_year	grade_level	...	first_9th_school_year_observed	temp2_first9year	temp3_first9year
2	2006	10	...	2003	2003	2003
2	2007	11	...	2003		2003

```
foreach grade in 10 11 12 {
    local numsubtract = `grade' - 9

    gen temp2_first9year = school_year - `numsubtract' if
first_9th_school_year_observed==. & first`grade'_flag==1

    by sid: egen temp3_first9year = min(temp2_first9year)

    replace first_9th_school_year_observed = temp3_first9year
if first_9th_school_year_observed==. & !mi(temp3_first9year)

    drop temp2_first9year temp3_first9year
}
```

// Review the distribution of first_9th_school_year_observed for students who transferred in grades 10-12

```
sort sid

tab first_9th_school_year_observed if observed_10==1 & observed_9==0 & sid!=sid[_n-1], mi

 first_9th_s |
chool_year_ |
   observed |      Freq.     Percent        Cum.
------------+-----------------------------------
       2005 |        409       26.20       26.20
       2006 |        322       20.63       46.83
       2007 |        384       24.60       71.43
       2008 |        446       28.57      100.00
------------+-----------------------------------
      Total |      1,561      100.00

tab first_9th_school_year_observed if observed_11==1 & observed_10==0 & observed_9==0 & sid!=sid[_n-1], mi

 first_9th_s |
chool_year_ |
   observed |      Freq.     Percent        Cum.
------------+-----------------------------------
       2004 |          2        0.22        0.22
       2005 |        288       32.25       32.47
       2006 |        285       31.91       64.39
       2007 |        318       35.61      100.00
------------+-----------------------------------
      Total |        893      100.00


tab first_9th_school_year_observed if observed_12==1 & observed_11==0 & observed_10==0 & observed_9==0 & sid!=sid[_n-1], mi

 first_9th_s |
chool_year_ |
   observed |      Freq.     Percent        Cum.
------------+-----------------------------------
       2004 |          2        0.70        0.70
       2005 |        137       48.24       48.94
       2006 |        145       51.06      100.00
------------+-----------------------------------
      Total |        284      100.00
```


/*** Step 4: Adjust the imputation of first_9th_school_year_observed for students who appear in a lower grade in a later school year. ***/

// 1. Flag students who are observed to be in a lower grade in a subsequent school year.

sid	school_year	grade_level	...	first_9th_school_year_observed	grade_flag	grade_flag_max
3	2006	10	...	2007	.	1
3	2007	9	...	2007	1	1

```
sort sid school_year

bys sid: gen grade_flag = 1 if grade_level[_n] < grade_
level[_n-1] & school_year[_n] > school_year[_n-1] & grade_
level >= 9 & grade_level <= 12

bys sid: egen grade_flag_max = max(grade_flag)
```

// 2. Flag the first school year in which students appear in high school grades

sid	school_year	grade_level	...	first_9th_school_year_observed	grade_flag	grade_flag_max	first_9th_flag
3	2006	10	...	2007	.	1	1
3	2007	9	...	2007	1	1	.

```
bys sid: gen first_9th_flag = _n == 1 if grade_level >=
9 & grade_level <= 12
```

sid	school_year	grade_level	...	first_9th_school_year_observed	grade_flag	grade_flag_max	first_9th_flag	temp4_first9year	temp5_first9year
3	2006	10	...	2007	.	1	1	2005	2005
3	2007	9	...	2007	1	1	.	.	2005

```
gen temp4_first9year = (school_year -
1) if grade_flag_max == 1 & first_9th_
flag == 1 & grade_level == 10

replace temp4_first9year = (school_
year - 2) if grade_flag_max == 1 &
first_9th_flag == 1 & grade_level ==
11

replace temp4_first9year = (school_year - 3) if grade_flag_max
== 1 & first_9th_flag == 1 & grade_level == 12

bys sid: egen temp5_first9year = min(temp4_first9year)
```

Task 3 IDENTIFYING THE NINTH-GRADE COHORT

// 4. Replace the first_9th_school_year_observed with the correctly imputed values.

sid	school_year	grade_level	...	first_9th_school_year_observed	...	temp5_first9year
3	2006	10	...	2005	...	2005
3	2007	9	...	2005	...	2005

```
replace first_9th_school_year_observed = temp5_first9year if
grade_flag_max == 1 & !mi(temp5_first9year)

sort sid

tab first_9th_school_year_observed if sid!=sid[_n-1]

 first_9th_s |
chool_year_ |
   observed |      Freq.     Percent        Cum.
------------+-----------------------------------
       2002 |          4        0.02        0.02
       2003 |          8        0.04        0.06
       2004 |          5        0.02        0.08
       2005 |      5,712       26.47       26.55
       2006 |      5,157       23.89       50.44
       2007 |      5,224       24.21       74.65
       2008 |      5,461       25.30       99.95
       2009 |         11        0.05      100.00
------------+-----------------------------------
      Total |     21,582      100.00
```

/*** Step 5: Keep only variables relevant to future analyses, and save the file. ***/

// Keep relevant variables

```
keep sid school_year grade_level frpl iep ell gifted total_days_enrolled total_days_absent days_suspended_out_of_school
first_9th_school_year_observed
```

// Save the current file as Student_School_Year_Ninth.dta.

```
save "${clean}\Student_School_Year_Ninth.dta", replace
```


Task 4 STUDENT SCHOOL ENROLLMENT

PURPOSE

In **Task 4**: Student School Enrollment, you will take the Student_School_Enrollment_Raw file and generate the Student_School_Enrollment file that matches the specification in **Identify**. After matching **Identify**, you will take your dataset a few steps further by consolidating overlapping enrollment spells and determining the last withdrawal code for each student to yield the file Student_School_Enrollment_Clean.

The core of this task:

1. Create a school_start and school_end variable.
2. Remove abnormal enrollment observations with missing enrollment and withdrawal dates along with enrollment or withdrawal dates that are not in the right order.
3. Consolidate overlapping enrollments by student by school.
4. Update days_enrolled based on the consolidated enrollments using the new enrollment and withdrawal dates.
5. Determine the last withdrawal code for each student. You will use this data in later analyses to determine a student's end of high school outcomes.

After completing this, you will have a clean Student_School_Enrollment file. This process sets up our analyses for high school graduation and college enrollment and persistence outcomes.

HOW TO START

To begin, open the Student_School_Enrollment_Raw file in Stata. If you do not have Stata, you can follow the steps of the task by looking at the instructions and data snippets we have provided.

If this is your first time attempting **Task 4**, start with the provided input file. This file teaches you SDP's cleaning methodology and allows you to check answers from a common dataset.

Task 4 STUDENT SCHOOL ENROLLMENT

DATA DESCRIPTION FOR RAW FILE

The input file in this case, Student_School_Enrollment_Raw, varies from Student_School_Enrollment in **Identify** in a number of ways. First, the data is not yet unique by **sid**, **school_year**, **school_code,** and **enrollment_date** as shown in **Identify**. For instance, some students may have identical enrollment dates associated with different withdrawal dates for the same school so the data is actually unique by **sid**, **school_year**, **school_code, enrollment_date**, and **withdrawal_date**. The aim of this task is to match Student_School_Enrollment in **Identify** in its structure and uniqueness so it is unique by **sid**, **school_year**, **school_code,** and **enrollment_date** and then take things one step further by consolidating any overlapping enrollment spells at the same school and defining the last withdrawal code. A second, more minor, difference is that **enrollment_code** and **withdrawal_code** are omitted: the **enrollment_code_desc** and **withdrawal_code_desc** are sufficient in this case.

Variable Name	Values or Data Type	Definition
sid	numeric	Student identifier unique to each student. This identification number is assigned to a student upon enrollment in your agency. State agencies may have different identification numbers than district agencies for the same student.
school_year	spring calendar year	Academic school year from fall to spring, denoted here as the spring calendar year.
school_code	use local values	The local numeric or alpha-numeric code for the school.
enrollment_date	date format (yyyy-mm-dd)	When the student enrolled at the school. Here enrollment dates are recorded at the beginning of each school year (e.g., 8th graders are assigned an enrollment date at the begining of the year even if they were enrolled at the same school the year before).
withdrawal_date	date format (yyyy-mm-dd)	The date the student withdrew from the school. Here withdrawal dates are recorded at the end of each school year (e.g., 8th graders are assigned a withdrawal date at the end of the school year even if they plan to attend the same school next year).
enrollment_code	use local values	The local numeric or alpha-numeric code describing enrollment reason into the school, if available.
enrollment_code_desc	text	Description of the enrollment_code.
withdrawal_code	use local values	The local numeric or alpha-numeric code that describes withdrawal reason from the school.
withdrawal_code_desc	text	Description of the withdrawal_code.
days_enrolled	number of days	Number of school days during the school year the student was enrolled at a given school. The system's data sources may report this directly, or you may calculate it based on enrollment data.
days_absent	number of days	Number of school days during the school year the student was marked absent at a given school. The system's data sources may report this directly, or you may calculate it based on enrollment data.

Uniqueness: Some students may have identical enrollment dates associated with different withdrawal dates for the same school. Additionally, some enrollment periods might overlap at the same school. If this occurs, enrollment should be consolidated into a single observation. To address both issues, you will take the raw file unique by **sid**, **school_year**, **school_code, enrollment_date**, and **withdrawal_date** and make it unique by **sid**, **school_year**, **school_code,** and **enrollment_date** with no overlapping same school enrollment.

Task 4 STUDENT SCHOOL ENROLLMENT

/*** Step 0: Load the Student_School_Enrollment_Raw data file ***/

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc	days_ enrolled
2	2006	486	27-Aug-05	Grade 10	5-Jun-06	Promoted End Year	282
2	2007	486	29-Aug-06	Grade 11	8-Mar-07	Retained in Grade	191
2	2007	486	1-Feb-07	Grade 11	5-Jun-07	Promoted End Year	124

```
use "${raw}\Student_School_Enrollment_Raw.dta",
clear
```

/*** Step 1: Create a school_start and school_end variable ***/

// In this example, school start is August 1, and school end is July 31 of each school year. This may be different in your agency.

```
gen school_start = mdy(8,1,school_year-1)
gen school_end = mdy(7,31,school_year)

format school_start school_end %td
```

/*** Step 2: Remove abnormal enrollment observations. ***/

// 1. Drop observations missing both enrollment and withdrawal dates.

```
drop if mi(enrollment_date) & mi(withdrawal_date)
```

// 2. Drop observations with enrollment and withdrawal dates on same day.

```
drop if enrollment_date == withdrawal_date & !mi(enrollment_date) & !mi(withdrawal_date)
```

// 3. Drop observations with withdrawal date earlier than enrollment date.

```
drop if withdrawal_date < enrollment_date & withdrawal_date!=.
```

// 4. Drop observations with enrollment date after the end of the current school year.

```
drop if enrollment_date >= school_end
```

// 5. Drop observations with enrollment date before the beginning of the current school year.

```
drop if enrollment_date < school_start
```

// 6. Drop observations with withdrawal date more than one month after the end of the school year.

```
drop if withdrawal_date > (school_end + 31) & !mi(withdrawal_date)
```

// 7. Check to make sure enrollment dates are in the correct school year.

```
assert enrollment_date >= school_start & enrollment_date <= school_end
```

/*** Step 3: Consolidate overlapping enrollments by student by school. ***/

// 1. Sort enrollment spells in ascending order and then check how many overlapping enrollment spells exist for a student at the same school.

```
sort sid school_code enrollment_date

count if sid==sid[_n-1] & school_code==school_code[_n-1] & enrollment_date <= withdrawal_date[_n-1] & withdrawal_
date[_n-1]!=.
```

```
710
```

// 2. For overlapping observations, replace the enrollment date and enrollment code description of all but the first enrollment spell with the earliest enrollment date

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc
2	2006	486	27-Aug-05	Grade 10	5-Jun-06	Promoted End Year
2	2007	486	29-Aug-06	Grade 11	8-Mar-07	Retained in Grade
2	2007	486	1-Feb-07	Grade 11	5-Jun-07	Promoted End Year

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc
2	2006	486	27-Aug-05	Grade 10	5-Jun-06	Promoted End Year
2	2007	486	29-Aug-06	Grade 11	8-Mar-07	Retained in Grade
2	2007	486	29-Aug-06	Grade 11	5-Jun-07	Promoted End Year

```
// Replace enrollment_date

replace enrollment_date = enrollment_date[_n-1] if
sid==sid[_n-1] & school_code==school_code[_n-1] &
enrollment_date <= withdrawal_date[_n-1] & with-
drawal_date[_n-1]!=.

// Replace enrollment_code_desc

replace enrollment_code_desc = enrollment_code_
desc[_n-1] if
sid==sid[_n-1] & school_code==school_code[_n-1] &
enrollment_date <= withdrawal_date[_n-1] & with-
drawal_date[_n-1]!=.
```


// 3. Replace the withdrawal date and withdrawal code description of the earliest enrollment spell with the latest withdrawal date.

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc
2	2007	486	27-Aug-05	Grade 10	5-Jun-07	Promoted End Year
2	2007	486	29-Aug-06	Grade 11	8-Mar-07	Retained in Grade
2	2006	486	29-Aug-06	Grade 11	5-Jun-06	Promoted End Year

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc
2	2007	486	27-Aug-05	Grade 10	5-Jun-07	Promoted End Year
2	2007	486	29-Aug-06	Grade 11	5-Jun-07	Promoted End Year
2	2006	486	29-Aug-06	Grade 11	5-Jun-07	Promoted End Year

```
// Sort the data first so that latest withdrawal
information appears as the first record.

gsort sid school_code enrollment_date -withdrawal_
date

// Replace withdrawal_date

replace withdrawal_date = withdrawal_date[_n-1] if
sid==sid[_n-1] & school_code==school_code[_n-1] &
enrollment_date==enrollment_date[_n-1]

// Replace withdrawal_code_description

replace withdrawal_code_desc = withdraw-
al_code_desc[_n-1] if sid==sid[_n-1] &
school_code==school_code[_n-1] & enrollment_
date==enrollment_date[_n-1]
```

/*** Step 4: Update days_enrolled based on the consolidated enrollments using the new enrollment and withdrawal dates. ***/

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc	days_ enrolled
2	2007	486	27-Aug-05	Grade 10	5-Jun-07	Promoted End Year	282
2	2007	486	27-Aug-05	Grade 10	5-Jun-07	Promoted End Year	280
2	2006	486	27-Aug-05	Grade 10	5-Jun-07	Promoted End Year	280

```
gen temp_days_enrolled = withdrawal_date - enroll-
ment_date

replace days_enrolled = temp_days_enrolled
drop temp_days_enrolled
```

/*** Step 5: Determine the last withdrawal code for each student. You will use this data in later analyses to determine a student's end of high school outcomes. ***/

Task 4 STUDENT SCHOOL ENROLLMENT

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc
16	2007	450	07-Jan-07	Grade11	5-Jun-07	Promoted End Year
16	2008	450	20-Aug-07	Grade 12	5-Jun-07	Graduated with Diploma

sid	school_ year	school_ code	enrollment_ date	enrollment_ code_desc	withdrawal_ date	withdrawal_ date_desc	temp_last_wd	last_ withdrawal_ reason
16	2008	450	20-Aug-07	Grade 12	5-Jun-07	Graduated with Diploma	Graduated with Diploma	Graduated with Diploma
16	2007	450	07-Jan-07	Grade11	5-Jun-07	Promoted End Year		Graduated with Diploma

```
gsort sid -withdrawal_date

bys sid: gen temp_last_wd = withdraw-
al_code_desc if _n==1

egen last_withdrawal_reason =
mode(temp_last_wd), by(sid)

drop temp_last_wd
```

/**** Step 6: Drop any unneeded variables, drop duplicates, and save the file ****/

// 1. Drop duplicate records

```
duplicates drop
```

// 2. Confirm that file is unique by student, school_year, school_code, and enrollment_date

```
isid sid school_year school_code enrollment_date
```

// 3. Save the current file as Student_School_Enrollment_Clean

```
save "${clean}\Student_School_Enrollment_Clean.dta", replace
```


Task 5 STUDENT TEST SCORES

PURPOSE

In **Task 5**: Student Test Scores, you will take the Student_Test_Scores file, containing data on all the tests a student has taken and matching the structure of **Identify**. Through this task, you will generate three different clean output files that contain a single score and test-taking instance for each student:

- Prior Achievement (one 8th grade state test score per student),
- SAT scores (one SAT score per student), and
- ACT scores (one ACT score per student).

The file for Prior Achievement will contain students' achievement on state standardized Math and English Language Arts tests in 8th grade. This will allow you to control for prior academic achievement when you examine college-going outcomes. The SAT and ACT score files will be used for defining highly qualified high school graduates.

The core of this task:

- Prior Achievement
  1. Clean state test scores and resolve instances where students took the same test multiple times.
  2. Standardize test scores to a mean of 0 and a standard deviation of 1. This allows you to compare across tests and years when different score scales were used.
  3. Generate a composite math and English score for scaled and standardized test scores in eighth grade.
- SAT
  1. Clean SAT test scores and resolve instances where students took the same test multiple times.
  2. Generate a total SAT score based on math, verbal, and writing scores.
- ACT
  1. Clean ACT test scores and resolve instances where students took the same test multiple times.

After completing this, you will have a Prior_Achievement file with 8th grade test scores. You will also have SAT and ACT files. All three files will be used in Connect.

HOW TO START

To begin, open the Student_Test_Scores file in Stata. This file contains data on State assessments, SAT, and ACT scores. If you do not have Stata, you can follow the steps of the task by looking at the instructions and data snippets we have provided.

If this is your first time attempting **Task 5**, start with the provided input file. This file teaches you SDP's cleaning methodology and allows you to check answers from a common dataset.

DATA DESCRIPTION

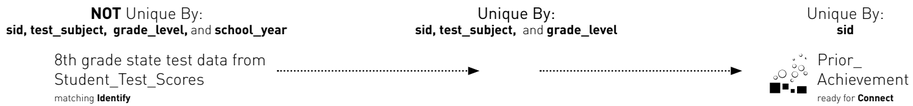
The input file, Student_Test_Scores, follows the structure of Student_Test_Scores in **Identify** so it is unique by **sid, test_code,** and **test_date**. The aim of this task will be to create three separate clean output files, Prior_Achievement, SAT, and ACT, that report only one test score per student. This means that for eight grade prior achievement duplicates of the same test taken in the same and different years will need to be resolved. Also any duplicates of SAT or ACT scores will need to be resolved as well.

Variable Name	Values or Data Type	Definition
sid	numeric	Student identifier unique to each student. This identification number is assigned to a student upon enrollment. State agencies may have different identification numbers than district agencies for the same student.
test_code	use local values	These values identify individual tests, (usually expressed as a sequence of letters and numbers). For example, a state test such as "MCAS 6th Grade Math in Massachusetts" or college entrance exam such as "SAT Math". Can be created from a concatenation of component variables below (e.g. test_type and test_subject), as it is in this case.
test_date	date format (yyyy-mm-dd)	The exact date (or at a minimum school year) the test was completed. Note that students who re-take tests or are retained may have multiple observations for a single test_code; these should be differentiated by test_date.
test_type	use local values	The category of test, e.g. MCAS (state test), SAT, ACT, or AP.
grade_level	-9 = ungraded -1 = any pre-kindergarten 0 = kindergarten 1-12 = grades 1–12 13+ = additional grade levels (i.e. vocational training, special education past year 12)	Numeric grade level of the test. May be unavailable for SAT, ACT, or AP tests.
test_subject	1 = math 2 = English language 3 = science 4 = social studies 5 = other 6 = writing	Subject of test.
language_version	E = English S = Spanish	Language of test.
raw_score	numeric	Student's raw score if available. May be unavailable for SAT, ACT, or AP tests.
scaled_score	numeric	Student's scaled score.
performance_level	code	Student's performance level (e.g., not proficient, proficient, advanced). May be unavailable for SAT, ACT, or AP tests.

Uniqueness:

Prior Achievement (8th grade state test scores),

Ideally, state test data in its raw form is unique by **sid, test_subject**, **grade_level,** and **school_year**. However, some students re-take the same test for the same grade in the same year. To fix this, you will make the 8th grade test score data in Student_Test_Scores unique by **sid, test_subject**, **grade_level,** and **school_year** by removing any same year repeat test taking instances. Then, you will manipulate the data so tests for different subjects in the same grade_level fall on the same row so the data is unique by **sid, test_subject**, and **grade_level**. As a final step, if a student took the same test in different years (e.g. by repeating a grade), you will take the earliest instance. The data will finally be unique by **sid** and is considered a clean file and ready to be incorporated into the analysis file in **Connect**.



SAT

Ideally, SAT test data in its raw form is unique by **sid**. However, some students re-take the SAT. To fix this, you will take the data unique by **sid, test_subject**, and **test_date** and reshape it so the data will finally be unique by **sid** and is considered a clean file and ready to be incorporated into the analysis file in **Connect**.

ACT

Ideally, ACT test data in its raw form is unique by **sid**. However, some students re-take the ACT. To fix this, you will take the data unique by **sid, test_subject**, and **test_date** and reshape it so the data will finally be unique by **sid** and is considered a clean file and ready to be incorporated into the analysis file in **Connect**.

Task 5 STUDENT TEST SCORES

/*** Part I: Clean Prior Achievement Scores ***/

// 0. Load the Student_Test_Scores data file.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
10	ACT	.	06-Apr-05	.	ACT Composite	14	.
10	State	2004	15-Apr-04	8	Math	729	37
10	State	2004	15-Apr-04	8	ELA	738	38
10	State	2005	15-Apr-05	9	Math	817	54
10	State	2007	15-Apr-05	9	ELA	814	50
10	State	2007	15-Apr-07	10	Math	563	14
10	State	2007	15-Apr-07	10	ELA	616	19

```
{
use "${raw}/Student_Test_Scores.dta", clear
```

// 1. Keep only the variables you need and limit the sample to state test scores in 8th grade.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
10	State	2004	15-Apr-04	8	Math	729	37
10	State	2004	15-Apr-04	8	ELA	738	38

```
keep sid test_type test_subject school_year grade_level
scaled_score raw_score test_date

keep if test_type=="State"
keep if grade_level==8
```

// 2. Clean up raw and scaled scores.

```
// Change raw and scaled scores to missing if zero.

foreach var in raw_score scaled_score {
      replace `var' = . if `var' == 0
}

//Drop observations missing both a raw and scaled test score.

drop if scaled_score==. & raw_score==.
```


Task 5 STUDENT TEST SCORES

// 3. Identify same-year repeat test takers and take the highest test score.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
595	State	2007	15-Apr-07	8	Math	770	45
595	State	2007	15-Apr-07	8	ELA	789	49
595	State	2007	15-Apr-07	8	ELA	799	50
sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
595	State	2007	15-Apr-07	8	Math	770	45
595	State	2007	15-Apr-07	8	ELA	799	50

```
gsort sid test_subject grade_level school_year -scaled_score

drop if sid==sid[_n-1] & test_subject==test_subject[_n-1]
& school_year==school_year[_n-1] & grade_level==grade_
level[_n-1] & scaled_score[_n-1]!=.

// Verify that each student has only one state test in a
subject in a school year.

isid sid test_subject grade_level school_year
```

// 4. Reshape the data so math and ELA tests appear on the same row.

sid	test_ type	school_ year	test_ date_ math	test_ date_ela	scaled_ score_ math	scaled_ score_ela	raw_ score_ math	raw_ score_ ela
595	State	2007	15-Apr-07	15-Apr-07	770	799	45	50

```
// Generate the stub names for math and
ELA

gen test_subject2 = "_math" if test_sub-
ject == 1
replace test_subject2 = "_ela" if test_
subject == 2
drop test_subject

reshape wide raw_score scaled_score test_date, i(sid test_
type school_year grade_level) j(test_subject2) string
```

// 5. Compute standardized test scores with mean 0 and standard deviation 1.

```
foreach subject in math ela {
      bys school_year grade_level: center scaled_score_`subject', standardize gen(scaled_`subject'_std)
}
```

// 6. Identify different-year repeat test takers and take the earliest test score.

sid	test_ type	school_ year	test_ date_ math	test_ date_ela	scaled_ score_ math	scaled_ score_ela	raw_ score_ math	raw_ score_ ela
8	State	2005	15-Apr-05	15-Apr-05	698	779	35	46
8	State	2006	15-Apr-06	15-Apr-06	754	676	40	32

sid	test_ type	school_ year	test_ date_ela	scaled_ score_ela	raw_ score_ ela
8	State	2005	15-Apr-05	779	46
8	State	2006	15-Apr-06	676	32

sid	test_ type	school_ year	test_ date_ela	scaled_ score_ela	raw_ score_ ela
8	State	2005	15-Apr-05	779	46

```
// First process ELA scores
preserve
      drop *_math*
      drop if scaled_score_ela==.

      // Keep only the earliest instance in which the stu-
dent took the test

      sort sid grade_level school_year
      drop if sid==sid[_n-1] & grade_level==grade_
level[_n-1] &  school_year>= school_year[_n-1] & scaled_
score_ela[_n-1]!=.

      bys sid: gen count=_n
      tab count
      drop count

      // Save the ela_scores as a tempfile to be merged on
      tempfile ela_scores

      save `ela_scores'
restore
```



```
// Next process math scores

drop *_ela*
drop if scaled_score_math==.
// Keep only the earliest instance in which the student took the test

sort sid grade_level school_year

drop if sid==sid[_n-1] & grade_   level==grade_level[_n-1] &
school_year>= school_year[_n-1] & scaled_score_math[_n-1]!=.

bys sid: gen count=_n
tab count
drop count

// Merge the ela_scores tempfile onto the math scores

merge 1:1 sid using `ela_scores', nogen
```

// 7. Verify that each student has only one state test, and drop unneeded variables.

```
isid sid
drop test_date_math test_date_ela test_type
```

// 8. Generate composite scaled and standardized scores that average ELA and math scores.

```
gen scaled_score_composite = (scaled_score_math + scaled_score_ela) / 2 if !mi(scaled_score_math) & !mi(scaled_score_ela)

gen scaled_score_composite_std = (scaled_math_std + scaled_ela_std) / 2 if !mi(scaled_math_std) & !mi(scaled_ela_std)
```

// 9. Save the current file as Prior_Achievement.dta.

```
order sid school_year grade_level
raw_score_math raw_score_ela scaled_ score_math scaled_score_ela scaled_score_composite
scaled_math_std scaled_ela_std scaled_score_composite_std

save "${clean}/Prior_Achievement.dta", replace

}
```

Task 5 STUDENT TEST SCORES

/*** Part II: Clean SAT Scores ***/

// 0. Load the Student_Test_Scores data file.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
366	SAT	.	28-Oct-07	.	Math	660	.
366	SAT	.	28-Oct-07	.	ELA	750	.
366	SAT	.	28-Oct-07	.	Writing	720	.
366	State	2004	15-Apr-04	8	Math	868	61
366	State	2004	15-Apr-04	8	ELA	827	54
366	State	2005	15-Apr-05	9	Math	1064	96
366	State	2005	15-Apr-05	9	ELA	969	76
366	State	2007	15-Apr-07	10	Math	672	32
366	State	2007	15-Apr-07	10	ELA	695	30

```
{
use "${raw}/Student_Test_Scores.dta", clear
```

// 1. Keep only the variables and limit the sample to SAT.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
366	SAT		28-Oct-07		Math	660	
366	SAT		28-Oct-07		ELA	750	
366	SAT		28-Oct-07		Writing	720	

```
keep if test_type == "SAT"
keep sid test_subject test_date scaled_score
```

// 2. Drop duplicate observations and any observations missing test scores.

```
duplicates drop
drop if mi(scaled_score)
```


Task 5 STUDENT TEST SCORES

// 3. Reshape the data so that math, ELA, and writing scores appear on one row by student and test date.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score1	scaled_ score2	scaled_ score6	raw_ score
366	SAT	.	28-Oct-07	.	Math	660	750	720	.

```
reshape wide scaled_score, i(sid test_date)
j(test_subject)
```

sid	test_ type	school_ year	sat_test_ date	grade_ level	test_ subject	sat_ math_ score	sat_ verbal_ score	sat_ writing_ score	raw_ score
366	SAT	.	28-Oct-07	.	Math	660	750	720	.

```
rename scaled_score1 sat_math_score
rename scaled_score2 sat_verbal_score
rename scaled_score6 sat_writing_score
rename test_date sat_test_date
```

// 4. Identify repeat test takers and take the earliest test score.

```
sort sid sat_test_date
bys sid: gen n = _n
tab n

keep if n==1
drop n

// Verify that the file is now unique by student.
isid sid
```

// 5. Verify that test scores from the component subjects are not missing and generate total scores.

```
assert !mi(sat_math_score) & !mi(sat_verbal_score)
gen sat_total_score = sat_math_score + sat_verbal_score
label var sat_total_score "combined math & verbal"

assert !mi(sat_math_score) & !mi(sat_verbal_score) & !mi(sat_writing_score)
gen sat_total_score_plus_writing = sat_math_score + sat_verbal_score + sat_writing_score
label var sat_total_score_plus_writing "combined math & verbal & writing"
```

// 6. Save the current file as SAT.dta.

```
save "${clean}\SAT", replace
}
```

Task 5 STUDENT TEST SCORES

/*** Part III: Clean ACT Scores ***/

// 0. Load the Student_Test_Scores data file.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
10	ACT	.	06-Apr-05	.	ACT Composite	14	.
10	State	2004	15-Apr-04	8	Math	729	37
10	State	2004	15-Apr-04	8	ELA	738	38
10	State	2005	15-Apr-05	9	Math	817	54
10	State	2007	15-Apr-05	9	ELA	814	50
10	State	2007	15-Apr-07	10	Math	563	14
10	State	2007	15-Apr-07	10	ELA	616	19

```
{
use "${raw}/Student_Test_Scores.dta", clear
```

// 1. Keep only the variables you need and limit the sample to ACT.

sid	test_ type	school_ year	test_ date	grade_ level	test_ subject	scaled_ score	raw_ score
10	ACT	.	06-Apr-05	.	ACT Composite	14	.

```
keep if test_type=="ACT"
keep sid test_subject test_date scaled_score
```

// 2. Identify repeat test takers and take the earliest test score.

```
sort sid test_date
bys sid: gen n = _n
tab n
keep if n==1
```


Task 5 STUDENT TEST SCORES

// 3. Keep and rename the relevant variables.

```
keep sid test_date scaled_score
rename test_date act_test_date
rename scaled_score act_composite_score

// Verify that the file is now unique by student.
isid sid
```

// 4. Save the current file as ACT.dta.

```
save "${clean}\ACT", replace

}
```

PURPOSE

In **Task 6**: Student Class Enrollment, you will take the Class_Raw file and the Student_Class_Enrollment file to create the Student_Class_Enrollment_Merged file that combines these two files together. The combined file will identify a unique observation by student and class id. To obtain this file, you will first clean the Class_Raw file to identify core courses in math and ELA based on the course description variable and match the specification in **Identify**. This will make the class file unique by class id. Second, you will merge the Class file and the Student Class Enrollment file and make it unique by student id and class id.

The core of this task:

1. Using the Class file:
    a. Drop incomplete observations
    b. Flag core math and English courses based on the course description
2. Merging the Student Class Enrollment file:
    a. Merge the Class file onto the Student_Class_Enrollment_Raw file
    b. Evaluate course marks and drop courses with no record of completion
    c. Evaluate course enrollment so that each student has only one enrollment record for a course

The Student_Class_Enrollment_Merged file will be used in **Connect** to create on-track indicators for students. On-track indicators explore year-by-year academic progress towards high school graduation and college readiness. For instance, using course credit and course grade information, one might ask what percent of students earn the minimum number of credits in their core courses to satisfy agency graduation requirements?

HOW TO START

To begin, open the Class_Raw file in Stata. This file contains data linking students to teachers. If you do not have Stata, you can follow the steps of the task by looking at the instructions and data snippets we have provided. In the second part of this task, you will then use the Student_Class_Enrollment file.

If this is your first time attempting **Task 6**, start with the provided input file. This file teaches you SDP's cleaning methodology and allows you to check answers from a common dataset.

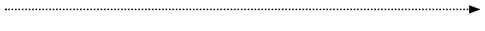
DATA DESCRIPTION FOR RAW FILE

The input file, Class_Raw, varies from Class in **Identify** in a number of key ways. Most importantly, the data is not unique by **cid** as shown in **Identify**. For instance, there may be more than one course description that describes the same course. Also, a **tid** is not included as it is not required for the questions later asked in this toolkit. Support for a Class file with **tid** will come with the Human Capital version of the toolkit. The aim of this task then is to eliminate any duplicate course code descriptions and match the Class file in **Identify** in its structure and uniqueness so it is unique by **cid** alone.

Field Name	Values or Data Type	Definition
cid	numeric	Variable that links students to teachers by grouping students in the same room at the same time. One unique value should be assigned to each combination of variables: school_year + school_code + course_code + section_code + period_bell + room_number + tid.
credits_possible	numeric	Credits possible from the course.
school_year	spring calendar year	Academic school year from fall to spring, denoted here as the spring calendar year.
school_code	use local values	The local numeric or alpha-numeric code for the school.
section_code		The local numeric or alpha-numeric code that identifies individual sections.
instructional_level		Instructional level of the course.
course_code_desc	string	Description of the course.
course_code	use local values	Course identifier.

Uniqueness:

NOT Unique By:
cid
Class_Raw
from **Identify**



Unique By:
cid
Class
matching **Identify**, ready for **Connect**

The input file, Student_Class_Enrollment_Raw, follows the structure of Student_Class_Enrollment in **Identify** so it is unique by **sid, cid,** and **class_enrollment_date**. The aim of this task then is to take things one step further by consolidating any overlapping enrollment spells for the same student and cid.

Field Name	Values or Data Type	Definition
sid	numeric	Student identifier unique to each student.
cid	numeric	Variable that links students to teachers by grouping students in the same room at the same time. One unique value should be assigned to each combination of variables: school_year + school_code + course_code + section_code + period_bell + room_number + tid.
class_enrollment_date	date format (yyyy-mm-dd)	The date the student enrolled in the class.
class_withdrawal_date	date format (yyyy-mm-dd)	The date the student withdrew from class.
marking_period	string	Indicates the length of the course. This could be year long, semester long, or quarterly. Use your agency values.
final_grade_mark	string	The final grade or mark the student received in the class.
final_grade_mark_num	numeric	The final grade or mark the student received in the class recorded as a numeric value.
credits_earned	numeric	Number of credits the student earned for the course.

Uniqueness:

Unique By:
sid, **cid**, and **class_enrollment_date**
Student_Class_Enrollment from **Identify**

Student_Class_Enrollment_Merged
matching and then furthering **Identify**, ready for **Connect**

Task 6 STUDENT CLASS ENROLLMENT

/*** Part I: Clean the Class file ***/

/*** Step 0: Load the Class_Raw data file. ***/

```
{
use "${raw}\Class_Raw.dta", clear

// 1. Identify the critical variables that identify a class.
local identifiers "cid school_year school_code section_code course_code" //sdptid

// 2. Drop the observations where any of the critical variables are missing
foreach var of local identifiers {
      drop if mi(`var')
}
```

/*** Step 2: Flag core math and English courses. ***/

```
// Note that agencies may have varying consistency in course
names and use different criteria to identify a core course
vs an elective.

// In some cases, other criteria may have to be applied to
identify core courses (e.g. the department the course is
listed in, or length of the course.)

// We provide a simplified version of the cleaning process
for the class file: work within your agency to determine the
best criteria.
```

// 1. Tabulate course names

```
tab course_code_desc
```

course_code_ desc	Freq.	Percent	Cum.
ALGEBRA	6,202	4.57	4.57
ALGEBRA II	21	0.02	4.59
CALCULUS	218	0.16	4.75
ELECTIVE I	27,468	20.26	25.01
ELECTIVE II	23,679	17.46	42.47
ELECTIVE III	23,702	17.48	59.95
ELECTIVE IV	32,184	23.74	83.69
ENG 10	101	0.07	83.76
ENGLISH 09	7,799	5.75	89.51
ENGLISH 10	177	0.13	89.64
ENGLISH 11	418	0.31	89.95
ENGLISH 12	287	0.21	90.16
ENGLISH 9	7,913	5.84	96.00
GEOM	2,444	1.80	97.80
GEOMETRY	2,409	1.78	99.58
OTHER ELA	151	0.11	99.69
OTHER MATH	310	0.23	99.92
STATISTICS	68	0.05	99.97
TRIGONOMETRY	44	0.03	100.00
Total	135,595	100.00	

Task 6 STUDENT CLASS ENROLLMENT

// 2. Flag math courses based on the tabulation results

```
// Generate a flag variable
gen math_flag = .

// Use the regexm function to identify course names that contain common word stems, but slightly different spellings, e.g.
Algebra I and Algebra-I

replace math_flag = 1 if regexm(course_code_desc, "GEOM") | regexm(course_code_desc, "MATH") | regexm(course_code_desc,
"ALGEBRA")

// Use the inlist funtion to identify other course names
replace math_flag = 1 if inlist(course_code_desc, "CALCULUS", "STATISTICS", "TRIGONOMETRY")

// Check the results of flagging your variables
tab course_code_desc math_flag, m
```

course_code_ desc	math_flag 1	.	Total
ALGEBRA	6,202	0	6,202
ALGEBRA II	21	0	21
CALCULUS	218	0	218
ELECTIVE I	0	27,468	27,468
ELECTIVE II	0	23,679	23,679
ELECTIVE III	0	23,702	23,702
ELECTIVE IV	0	32,184	32,184
ENG 10	0	101	101
ENGLISH 09	0	7,799	7,799
ENGLISH 10	0	177	177
ENGLISH 11	0	418	418
ENGLISH 12	0	287	287
ENGLISH 9	0	7,913	7,913
GEOM	2,444	0	2,444
GEOMETRY	2,409	0	2,409
OTHER ELA	0	151	151
OTHER MATH	310	0	310
STATISTICS	68	0	68
TRIGONOMETRY	44	0	44
Total	11,716	123,879	135,595

```
// Once each math course has been flagged, mark the non-math courses.
replace math_flag = 0 if math_flag ==.
```

// 3. Repeat this process for flagging ELA courses

```
// Generate a flag variable
gen ela_flag = .

// Use the regexm function to identify course names that contain common word stems, but slightly different spellings
replace ela_flag = 1 if regexm(course_code_desc, "ENG") | regexm(course_code_desc, "ELA") | regexm(course_code_desc, "OTHER
ELA")

// Check the results of flagging your variables
tab course_code_desc ela_flag, m
```

course_code_ desc	ela_flag 1	.	Total
ALGEBRA	0	6,202	6,202
ALGEBRA II	0	21	21
CALCULUS	0	218	218
ELECTIVE I	0	27,468	27,468
ELECTIVE II	0	23,679	23,679
ELECTIVE III	0	23,702	23,702
ELECTIVE IV	0	32,184	32,184
ENG 10	101	0	101
ENGLISH 09	7,799	0	7,799
ENGLISH 10	177	0	177
ENGLISH 11	418	0	418
ENGLISH 12	287	0	287
ENGLISH 9	7,913	0	7,913
GEOM	0	2,444	2,444
GEOMETRY	0	2,409	2,409
OTHER ELA	151	0	151
OTHER MATH	0	310	310
STATISTICS	0	68	68
TRIGONOMETRY	0	44	44
Total	16,846	118,749	135,595

```
// Once each ELA course has been flagged, mark the non-ELA courses.
replace ela_flag = 0 if ela_flag ==.
```


/**** Step 3: Drop any unneeded variables, drop duplicates, and save the temporary file ****/

// 1. Drop the course_code_desc, as it is no longer needed.

```
drop course_code_desc
duplicates drop
```

// 2. Verify that the data is unique by cid, and also unique by school year, school code, section code and course code.

```
isid cid
isid school_year school_code section_code course_code
```

// 3. Save the data in a temporary file

```
tempfile class
save `class'
}
```

/*** Part II: Clean the Student_Class_Enrollment file. ***/

/*** Step 0: Load the Student_Class_Enrollment data file. ***/

```
{
use "${raw}\Student_Class_Enrollment.dta", clear
```

/*** Step 1: Merge on the temporary Class file you saved earlier to the Student_Class_Enrollment file ***/

```
merge m:1 cid using `class'

// Keep only observations that merged from both files
keep if _merge==3
drop _merge
```

/*** Step 2: Evaluate course marks. ***/

```
tab final_grade_mark credits_possible, m
tab final_grade_mark final_grade_mark_num, m
// Some letter marks (NGPA and P) indicate that they do not count toward GPA, so you may leave the numeric mark as miss-
ing.
```

/*** Step 3: Evaluate course completion. ***/

```
// Drop observations that have no record of course completion
drop if mi(final_grade_mark) & mi(final_grade_mark_num) & mi(credits_earned)
```

/*** Step 4: Evaluate course enrollment. ***/

// Fix cases where a student has multiple observations for the same course with the same year and marking period (i.e. with overlapping enrollment dates)

// 1. Remove enrollment and withdrawal dates that are not in the current school year.

sid	cid	school_ code	school_ year	marking_ period	section_ code
2251	78150780	540	2006	51	7
2251	78150780	540	2006	51	7
2251	78150780	540	2006	51	7
2251	78150780	540	2006	51	7

sid	cid	school_ code	school_ year	marking_ period	section_ code	class_ enrollment_date	class_ withdrawal_date
2251	78150780	540	2006	51	7	12-Aug-05	17-Aug-05
2251	78150780	540	2006	51	7	13-Sep-05	27-Aug-05
2251	78150780	540	2006	51	7	21-Sep-05	2-Nov-05
2251	78150780	540	2006	51	7	23-Dec-05	10-Aug-06

```
foreach var in enrollment withdrawal {
    replace class_`var'_date = . if class_`var'_
date < mdy(8,1,school_year-1) | class_`var'_date >
mdy(7,31,school_year)
}
```

// 2. Identify the variables that identify a course

```
local identifier "sid cid school_year marking_period"
```


// 3. Populate all enrollments with the earliest enrollment date

sid	cid	school_ code	school_ year	marking_ period	section_ code	class_ enrollment_date	class_ withdrawal_date	temp_ first_enroll	first_ enroll
2251	78150780	540	2006	51	7	12-Aug-05	17-Aug-05	12-Aug-05	12-Aug-05
2251	78150780	540	2006	51	7	13-Sep-05	27-Aug-05	.	12-Aug-05
2251	78150780	540	2006	51	7	21-Sep-05	2-Nov-05	.	12-Aug-05
2251	78150780	540	2006	51	7	23-Dec-05	10-Aug-06	.	12-Aug-05

```
sort `identifier' class_enrollment_date
bys `identifier': gen temp_first_enroll = class_enrollment_date if _n==1
egen first_enroll = max(temp_first_enroll), by(`identifier')
format first_enroll %td
```

sid	cid	school_ code	school_ year	marking_ period	section_ code	class_ withdrawal_date	class_ enrollment_ date
2251	78150780	540	2006	51	7	17-Aug-05	12-Aug-05
2251	78150780	540	2006	51	7	27-Aug-05	12-Aug-05
2251	78150780	540	2006	51	7	2-Nov-05	12-Aug-05
2251	78150780	540	2006	51	7	10-Aug-06	12-Aug-05

```
drop temp class_enrollment_date
rename first_enroll class_enrollment_date
```

// 4. Populate all enrollments with the latest withdrawal date

sid	cid	school_ code	school_ year	marking_ period	section_ code	class_ enrollment_date	class_ withdrawal_date	temp_ first_enroll	first_ enroll	temp_last_ withdraw	last_ withdraw
2251	78150780	540	2006	51	7	12-Aug-05	17-Aug-05	12-Aug-05	12-Aug-05		2-Nov-05
2251	78150780	540	2006	51	7	13-Sep-05	27-Aug-05	.	12-Aug-05		2-Nov-05
2251	78150780	540	2006	51	7	21-Sep-05	2-Nov-05	.	12-Aug-05	2-Nov-05	2-Nov-05
2251	78150780	540	2006	51	7	23-Dec-05	10-Aug-06	.	12-Aug-05		2-Nov-05

```
gsort `identifier' -class_withdrawal_date
bys `identifier': gen temp_last_withdraw = class_withdrawal_date if _n==1
egen last_withdraw = max(temp_last_withdraw), by(`identifier')

format last_withdraw %td
drop temp class_withdrawal_date
rename last_withdraw class_withdrawal_date
```

/**** Step 5: Drop any unneeded variables, drop duplicates, and save the file ****/

// 1. Drop duplicate values

```
duplicates drop
```

// 2. Verify that the file is unique by sid and cid

```
isid sid cid
```

// 3. Order the variables

```
order sid cid school_year school_code course_code marking_period section_code instructional_level credits_possible math_
flag ela_flag class_enrollment_date class_withdrawal_date final_grade_mark final_grade_mark_num credits_earned
```

// 4. Sort the data

```
sort sid school_year marking_period cid
```

// 5. Save the current file as Student_Class_Enrollment_Merged.dta.

```
save "${clean}\Student_Class_Enrollment_Merged.dta" , replace
}
```


PURPOSE

In **Task 7**: Student NSC Enrollment, you will take the Student_NSC_Enrollment file that matches the specification in **Identify** and produce a Student_NSC_Enrollment_Indicators file that includes some of the first college enrollment indicators you will need for further analysis.

College enrollment data is obtained from the National Student Clearninghouse (NSC). NSC matches students from a file your agency sends, including student id, student name, high school from where the student graduated, graduation date, and some other variables. For more information on the NSC matching process and requirements, visit http://www.studentclearinghouse.org/high_schools/studenttracker

The core of this task:

1. Rename the variables typically returned by NSC
2. Format the date values
3. Standardize the variables that reflect the type of college the student enrolls in
4. Create a college graduation indicator
5. Interpret the college enrollment status
6. Identify the first college the student attended

After this task, you will merge the Student_NSC_Indicators file onto the college-going analysis file from **Connect**. You will use this file and the high school graduation variables you will also create in **Connect** to then to generate further college-going variables, such as variables that indicating if a student enrolled in college the fall after graduation, enrolled in college a year after graduation, and persisted through subsequent years of college.

HOW TO START

To begin, open the Student_NSC_Enrollment file in Stata. This file contains data on college enrollment and persistence for students in your agency. If you do not have Stata, you can follow the steps of the task by looking at the instructions and data snippets we have provided.

If this is your first time attempting **Task 7**, start with the provided input file. This file teaches you SDP's cleaning methodology and allows you to check answers from a common dataset.

Task 7 STUDENT NSC ENROLLMENT

DATA DESCRIPTION

The input file, Student_NSC_Enrollment, follows the structure of Student_NSC_Enrollment in **Identify** so it is unique by **sid, college_code_branch, enrollment_begin**, and **enrollment_end**. This usually equates to a semester. Though the exact structure of the data you receive from NSC may vary, it will likely look something like this. The aim of this task then is to become familiar with the NSC data and start building college enrollment outcomes that will be expanded upon in **Connect**.

Field Name	Values or Data Type	Definition
sid	numeric	Student identifier unique to each student. This identifier aligns with the student identifer that has been used in all other files, though it may not initially be in that form coming directly from NSC.
college_code_branch	string	OPE/FICE code of the college that the student attended. This is usually a six-digit college code followed by a hyphen and a two-digit branch code.
enrollment_begin	YYYYMMDD	Start of enrollment.
enrollment_end	YYYYMMDD	End of enrollment.
record_found_yn	"Y" "N"	Whether or not the NSC has college enrollment data for the student.
college_name	string	Name of the post-secondary institution
college_state	string	State in which the postsecondary institution that the student attended is located. The common USPS state abbreviations are used
year4year	"4-year" "2-year" "Less Than 2 Years"	Indicates the type of post-secondary institution
public_private	"Public" "Private"	Indicates the type of post-secondary institution
enrollment_status	"A" = leave of absence "D" = deceased "F" = full time "H" = half time "L" = less than half time "W" = withdrew	The last enrollment status reported for the student. If the reporting institution hasn't defined the student's enrollment status as directory information, this field will be left blank.
graduated	"Y" "N"	Graduation status information available from the reporting institution
graduation_date	YYYYMMDD	Date of student's graduation or degree achievement as provided by the reporting postsecondary institution
college_sequence	numeric	The sequential order of each school that the student attended. The enrollment records from the first school that the student attended will have a "1" in this field, enrollment records from the second school attended will have a "2" in this field, and so on.
degree_title	string	If available, the title of the degree the student received as provided by the reporting institution
major	string	If available, the major associated with the student's degree as provided by the reporting institution

Uniqueness: This dataset matches the specification in **Identify** and is unique by **sid, college_code_branch, enrollment_begin**, and **enrollment_end.**

Unique By:
sid, college_code_branch, enrollment_begin, and **enrollment_end**
(usually consitituting a semester)

Student_NSC_Enrollment
matching **Identify**

Task 7 STUDENT NSC ENROLLMENT

/*** Step 0: Load the Student_NSC_Enrollment data file. ***/

sid	record_ found_yn	enroll-ment_be-gin	enrollment_ end	college_code_ branch	college_ name	yr2_ yr4	public_ private	enroll-ment_ status	graduated	graduation _date	college_ sequence
13047	Y	20090110	20090505	164039-00	NY	2-year	Public	F	N		1
13047	Y	20080830	20081217	416739-00	TX	4-year	Private	F	N		1
13047	Y	20090829	20091215	164039-00	NY	2-year	Public	H	N		1
13047	Y	20080830	20081216	164039-00	NY	2-year	Public	H	N		1

```
use "${raw}/Student_
NSC_Enrollment.dta",
clear
```

// 1. Rename variables to indicate that they are NSC variables.

sid	n_record_ found_yn	n_enrollment_ begin	n_ enrollment_ end	n_college_ code_branch	n_college_ name	yr2_ yr4	public_ private	n_enrollment_ status	graduated	n_degree _date	n_enrl_ sequence
13047	Y	20090110	20090505	164039-00	NY	2-year	Public	F	N		1
13047	Y	20080830	20081217	416739-00	TX	4-year	Private	F	N		1
13047	Y	20090829	20091215	164039-00	NY	2-year	Public	H	N		1
13047	Y	20080830	20081216	164039-00	NY	2-year	Public	H	N		1

```
rename record_found_yn        n_record_found_yn
rename college_code_branch    n_college_opeid
rename college_name           n_college_name
rename degree_title           n_degree_title
rename major                  n_degree_major
rename college_sequence       n_enrl_sequence
rename enrollment_begin       n_enroll_begin_date
rename enrollment_end         n_enroll_end_date
rename graduation_date        n_degree_date
```

// 2. Format the date values as dates.

```
foreach var of varlist n_enroll_begin_date  n_enroll_end_date  n_degree_date {
      replace `var' = date(string(`var' ,"%08.0f"),"YMD")
      format %td `var'
}
```

// 3. Standardize types of college by:

```
// 2-year and 4-year college

tab yr2_yr4

        yr2_yr4 |      Freq.     Percent        Cum.
----------------+-----------------------------------
          2-year |      3,772       34.62       34.62
          4-year |      7,117       65.31       99.93
Less Than 2 Years |          8        0.07      100.00
----------------+-----------------------------------
           Total |     10,924      100.00

gen n_college_4yr = (yr2_yr4 == "4-year")
gen n_college_2yr = (yr2_yr4 == "2-year") | (yr2_yr4 == "Less Than 2 Years")
drop yr2_yr4

// Public and private college
tab public_private

public_priv |
        ate |      Freq.     Percent        Cum.
------------+-----------------------------------
    Private |      2,660       24.41       24.41
     Public |      8,237       75.59      100.00
------------+-----------------------------------
      Total |     10,897      100.00

gen n_college_public = (public_private == "Public")
gen n_college_private = (public_private == "Private")
drop public_private

// In-state and out-of-state college
tab college_state

college_sta |
         te |      Freq.     Percent        Cum.
------------+-----------------------------------
         CA |      1,915       17.57       17.57
         FL |      1,905       17.48       35.06
         IL |      1,683       15.44       50.50
         MA |      1,865       17.11       67.61
         NY |      1,779       16.33       83.94
         TX |      1,750       16.06      100.00
------------+-----------------------------------
      Total |     10,897      100.00
```



```
gen n_college_instate = (college_state == "$mystate")
gen n_college_outstate = (college_state != "$mystate") & !mi(college_state)
drop college_state
```

// 4. Create a college graduation indicator.

```
gen n_degree = (graduated == "Y")
drop graduated
```

// 5. Interpret enrollment status.

```
tab enrollment_status, m

enrollment_ |
     status |      Freq.     Percent        Cum.
------------+-----------------------------------
            |      1,118        9.33        9.33
          F |      8,693       72.53       81.86
          H |      1,350       11.26       93.12
          L |        551        4.60       97.72
          W |        273        2.27      100.00
------------+-----------------------------------
      Total |     11,895      100.00

gen n_enrl_status = .
// full time
replace n_enrl_status = 1 if enrollment_status == "F"
// half time
replace n_enrl_status = 2 if enrollment_status == "H"
// less than half time
replace n_enrl_status = 3 if enrollment_status == "L"
// withdrew
replace n_enrl_status = 4 if enrollment_status == "W"
// leave of absence
replace n_enrl_status = 5 if enrollment_status == "A"
// deceased
replace n_enrl_status = 6 if enrollment_status == "D"

label define n_enrl_status 1 "Full-Time" 2 "Half-Time" 3 "Less than Half-Time" 4 "Withdrew" 5 "Leave of Absence" 6 "De-
ceased", replace
label values n_enrl_status n_enrl_status
drop enrollment_status
```

/*** Step 2: Identify first college attended by type (any, 4-year and 2-year) that didn't result in a withdrawal. ***/

// 1. Specify these types of college (any, 4-year, 2-year) in globals.

```
global condition_any "n_college_4yr == 1 | n_college_2yr == 1"
global condition_4yr "n_college_4yr == 1"
global condition_2yr "n_college_2yr == 1"
```

// 2. Calculate the days enrolled.

```
gen days_enrolled = n_enroll_end_date - n_enroll_begin_date
```

// 3. Identify the first college a student enrolled in by type (any, 2-year, and 4-year).

```
foreach type in any 2yr 4yr {

    // Identify the first enrollment date a student is enrolled full time, half time or less than half time (based on n_
enrl_status)

    bys sid: egen first_enrl_date_`type' = min(n_enroll_begin_date) if ${condition_`type'} & (n_enrl_status <= 3 | n_enrl_
status == .)
    format first_enrl_date_`type' %td

    // Identify the name and ID of the first college of each type
foreach var in name opeid {
    // Get the  college name and id for the first enrollment date

    gen temp_first_college_`var'_`type' = n_college_`var' if n_enroll_begin_date == first_enrl_date_`type' & first_enrl_
date_`type' != .

    // Count how many first college names and ids you got for each student

    bys sid: egen nvals_first_college_`var'_`type' = nvals(temp_first_college_`var'_`type')
    // If a student started at multiple colleges of the same type on the same date, indicate this by replacing
    // these values with a dummy value (">1") for processing.
    replace temp_first_college_`var'_`type' = ">1" if nvals_first_college_`var'_`type' > 1 & nvals_first_college_`var'_`type'
!=.
```



```
      // For students who end up with multiple colleges, apply the following selection order:
      // (1) take the one with the highest enrollment status (full-time, half-time, less than half time in order of impor-
tance) then (2) longest days enrolled.

      gsort sid  n_enroll_begin_date  n_enrl_status  -days_enrolled
      bys sid: replace temp_first_college_`var'_`type' = n_college_`var' if _n==1 & ///
      nvals_first_college_`var'_`type' > 1 & nvals_first_college_`var'_`type' !=.

      // Remove the remaining ">1" values.
      replace temp_first_college_`var'_`type' = "" if temp_first_college_`var'_`type' == ">1"

      // Assign the first college
      egen first_college_`var'_`type' = mode(temp_first_college_`var'_`type'), maxmode by(sid)

      }
}
```

/**** Step 3: Drop any unneeded variables, and save the file ****/

// 1. Drop the unneeded variables

```
drop temp* nvals* days_enrolled
```

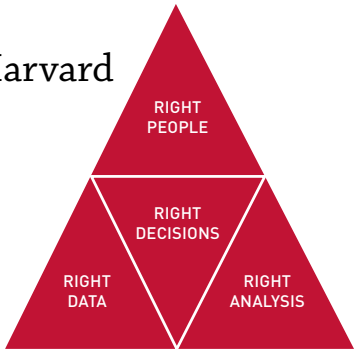
// 2. Save the current file as Student_NSC_Enrollment_Indicators

```
save "${clean}\Student_NSC_Enrollment_Indicators.dta", replace
```

The Strategic Data Project

OVERVIEW

The Strategic Data Project (SDP), housed at the Center for Education Policy Research at Harvard University, partners with school districts, school networks, and state agencies across the US. **Our mission is to transform the use of data in education to improve student achievement.** We believe that with the right people, the right data, and the right analyses, we can improve the quality of strategic policy and management decisions.



SDP AT A GLANCE

23 AGENCY PARTNERS
14 SCHOOL DISTRICTS
7 STATE EDUCATION DEPARTMENTS
2 CHARTER SCHOOL ORGANIZATIONS

79 FELLOWS
54 CURRENT
25 ALUMNI

CORE STRATEGIES

1. Placing and supporting top-notch analytic leaders as "Fellows"for two years with our partner agencies

2. Conducting rigorous diagnostic analyses of teacher effectiveness and college-going success using existing agency data

3. Disseminating our tools, methods, and lessons learned to many more education agencies

SDP DIAGNOSTICS

SDP's second core strategy, conducting rigorous diagnostic analyses using existing agency data, focuses on two core areas: (1) college-going success and attainment for students and (2) human capital (primarily examining teacher effectiveness).

The diagnostics are a set of analyses that frame actionable questions for education leaders. By asking questions such as, "How well do students transition to postsecondary education?" or "How successfully is an agency recruiting effective teachers?" we support education leaders to develop a deep understanding of student achievement in their agency.

ABOUT THE SDP TOOLKIT FOR EFFECTIVE DATA USE

SDP's third core strategy is to disseminate our tools, methods, and lessons learned to many more educational agencies. This toolkit is meant to help analysts in all educational agencies collect data and produce meaningful analyses in the areas of college-going success and teacher effectiveness. Notably, the analyses in this release of our toolkit primarily support questions related to college-going success. The data collection (Identify) and best practices (Adopt) stages of the toolkit, however, are applicable to any sort of diagnostic and convey general data use guidelines valuable to any analysts interested in increasing the quality and rigor of their analyses. Later releases will address analyses relating to teacher effectiveness.

Center for Education Policy Research
HARVARD UNIVERSITY

CENTER FOR EDUCATION POLICY RESEARCH
STRATEGIC DATA PROJECT
50 CHURCH ST., 4TH FLOOR, CAMBRIDGE, MA 02138
VOX 617.496.1563
FAX 617.495.2614
WWW.GSE.HARVARD.EDU/SDP