Database Design

Client: United Way of Lehigh Valley

Team SQ(L)uad
Agathe Benichou, Frankie Klerer, Nick Turney, Liam Mungovan

October 24th 2016

Our organization was approached by the United Way of the Greater Lehigh Valley to come up with an improvement of their current system. The issue with the existing system is that the data used assess schools is organized in different formats and it is not easy to read or query information from. The objective for our organization is to merge and normalize the data so that United Way can efficiently query and analyze trends. This will assist the United Way in achieving their goal of ensuring the academic success of all students across the Lehigh Valley. To address this problem, we have created a relational database that will store all the information regarding the academic performance of schools across the state of Pennsylvania. This design document describes the organization of our database and the reasons behind our design decisions.

In this design document, we have a basic format for each of the schemas we have made pertaining to the data given by the Pennsylvania Department of Education. For each relation, we have a high level description, a list of their attributes included, and an explanation of their relation to each other. Also included in our design document are the schemas for the Data Dictionary, the Sources and the Version Control. The Data Dictionary unifies and clarifies the columns of information given by the PA Dept of Education. The Sources tables organizes the sources that our data comes from. The Version Control table allows us to track any changes we make to our schema.

Schema of Relations:

Data Dictionary:

The data dictionary defines both the modifies as well as the original source names. The table is designed to clarify the meaning behind each of the attributes and assist anyone trying to compare the table to the original source. This table also helps specify how how each attribute in our database is used. Here is the schema of our table:

```
CREATE TABLE edprojectattributes (
        sourceid character varying(80),
        sourceattributename character varying(80),
        attributedescription character varying(160),
        relationname character varying(100),
        domain character varying(100),
        classattributename character varying(100)
);
```

- The data dictionary is too large to be inserted in this document so it's schema and relations can be found here: http://dev5.listyourself.net/relationscs320/ along with the schemas for the other relations.

Sources:

We added a Sources table which documents the sources for our information. This table allows any database user to find the original source of the data, as is standard for any commercial database. The table includes the original source name as well as any abbreviations used in the database. In addition, a data accessed attribute is included so the user knows when the data was last updated. Here is the schema for our table:

```
CREATE TABLE sources (
        sourceid integer,
        sourceorganization character varying(200),
        shortname character varying(100),
        sourcename character varying(200),
        sourceurl character varying(500),
        lastaccessed character varying(100)
);
```

Sources Relation:

sourceid	sourceorganization	shortname	sourcename	sourceurl	lastaccessed
1	Pennsylvania Department Of Education	PAED Grad Data	Pennsylvania Department Of Education Gradutate Data and Statistics	www.education.pa.gov/Data-and-Statistics/Pages/Graduates.aspx#tab-1	10/22/2016
2	Pennsylvania Department Of Education	PAED Cohort Grad Rate	Pennsylvania Department Of Education Cohort Graduation Rate	www.education.pa.gov/Data-and-Statistics/Pages/Cohort-Graduation-Rate-.aspx#tab-1	10/22/2016

In addition to the data dictionary and the source table, we have included a Version Control table to keep track of our table schemas as they are changed. This table will help keep track of our progress as we work. This simple table has attributes for a release number and the date each version was released. Here is the schema for our table:

```
CREATE TABLE versioncontrol (
        versionnumber integer,
        releasedate date
);
```

Version Control Relation:

versionnumber	releasedate

Schema of School:

The School table is the first table which relates to our team's data which organizes different rates by school. This table lists all the schools in Pennsylvania whose data has been tracked. Its attributes are explained further in the document.

```
CREATE TABLE school (
    school_id integer NOT NULL,
    aun integer,
    school_name character varying(200),
    county character varying(200),
    type character varying(20),
    lea character varying(75)
);
```

School Relation:

school_id	aun	school_name	county	type	lea

Schema of Multi Year Rates:

The multiyearrates table groups the first chunk of data: it compares the graduation rates of different genders, ethnicities and socioeconomic background of schools.

```
CREATE TABLE multiyearrates (
    schoolid integer NOT NULL,
    startyear character varying(10) NOT NULL,
    timeperiod integer NOT NULL,
    totalcohort character varying(20),
    totalgrads character varying(20),
    male double precision,
    female double precision,
    white double precision,
```

```
    hispanic double precision,
    black double precision,
    asian double precision,
    multiracial double precision,
    americanindianalaskan double precision,
    specialed double precision,
    econdisadv double precision,
    migrant double precision,
    ell double precision,
    nativehawaiianpacificislander double precision,
    endyear character varying(10),
);
```

Multi year rates Relation:

schoolid	startyear	timeperiod	totalcohort	totalgrads	male	female	white	hispanic	black	asian	multiracial	americanindianalaskan	specialed	econdisadv	migrant	ell	nativehawaiianpacificislander	endyear

Schema of Yearly Rates:

The yearlyrates table groups the second chunk of data: it compares the rates of where graduates went after graduation.

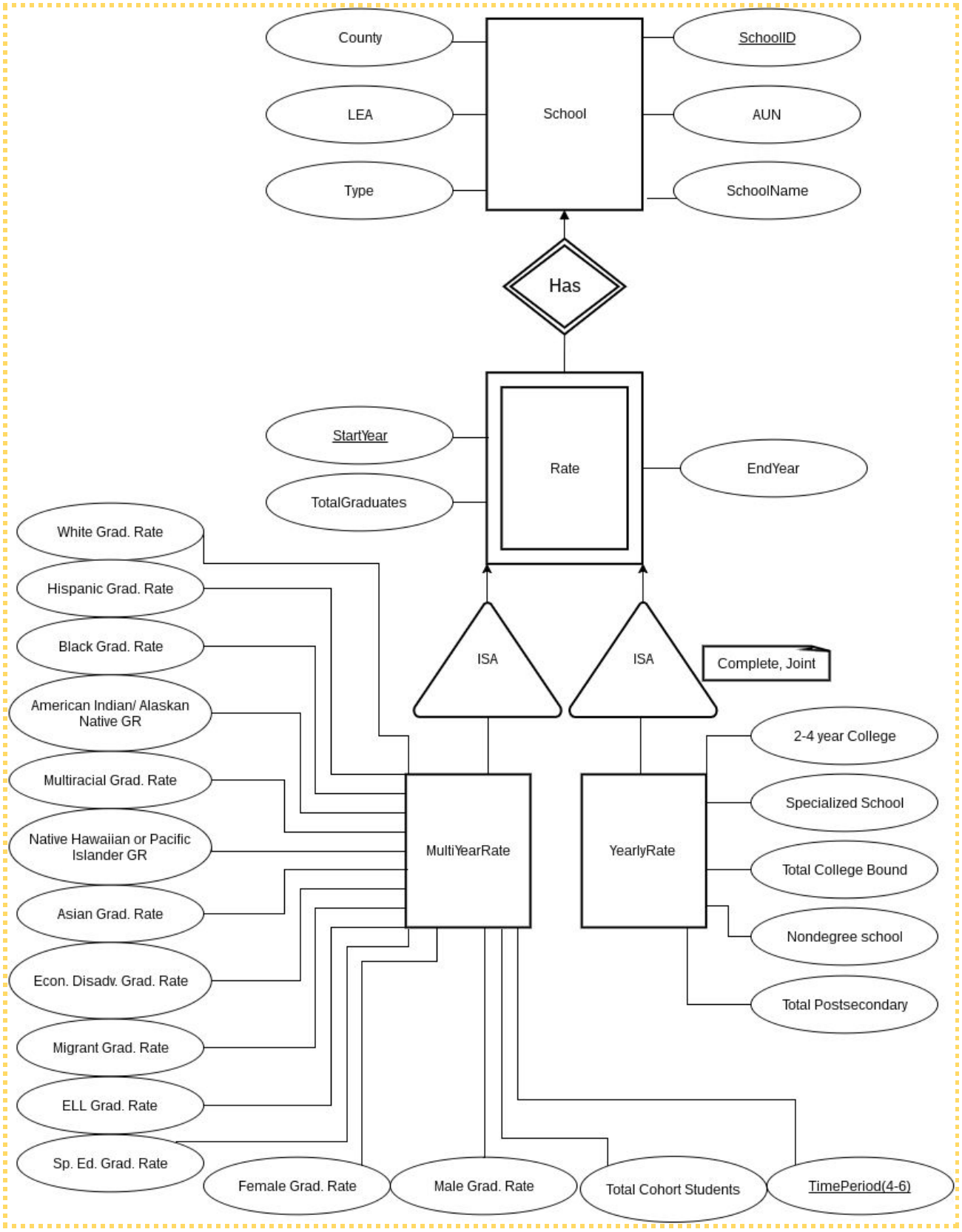
```
CREATE TABLE yearlyrates (
    schoolid integer NOT NULL,
    startyear integer NOT NULL,
    totalgrads character varying(20),
    twofouryearcollegepercent double precision,
    specializeddegreepercent double precision,
    totalcollegeboundpercent double precision,
    totalndgpsspercent double precision,
    totalpostsecondarypercent double precision,
    endyear character varying(10),
);
```

Yearly Rates Relation:

schoolid	startyear	totalgrads	timeperiod	twofouryearcollegepercent	specializeddegreepercent	totalcollegeboundpercent	totalndgpsspercent	totalpostsecondarypercent	endyear

We created the relations for our data that we have from our sources. We have 3 relations: School, Multi Year Rates and Yearly Rates. Each are shown and explained on following pages.

HIGH LEVEL DATABASE DESIGN



Derived Relations:

Our data comes from a single source, the Pennsylvania Department of Education which records and reviews statistical information regarding schools and student performance across the state. The data we are analyzing are the Cohort Graduation Rates and the Graduate Data and Statistics. The Cohort Graduation Rate data "compiles statistical information covering the 4-Year, 5-Year, and 6-Year Graduation Rates for Pennsylvania public schools. The cohort graduation rates are a calculation of the percentage of students who have graduated with a regular high school diploma within a designated number of years since the student first entered high school." The Graduate Data "reports provide information on intended post-high school activity of graduates, including college bound students" (PA Dep. of Edu.). By merging the information from these two sources together, a more complete report of a school's performance is revealed. This data compares two different kinds of information but it is brought together in a single tuple.

- School(School_ID, AUN, Name, Type, County, LEA)

Derived from the School entity set in the the E/R diagram. A School supports every tuple of data in our dataset. The key to the School relation is ID, or 'School ID' and 'School Code' in our datasets, which is a unique 4-digit number assigned by the state of Pennsylvania to every school entity. The School relation holds all persistent information on a school. This relation links together the other two relations: every school has different graduation rates for gender, race, and socioeconomic backgrounds and the rates for the graduates going to different kinds of post graduation schools.

- MultiYearRates(schoolid, startyear, timeperiod, totalcohort, totalgrads, male, female, white, hispanic, black, asian, multiracial, americanindianalaskan, specialed, econdisadv, migrant, ell, nativehawaiinpacificislander, endyear)

This relation compares the graduation rates of students from different races, genders and socioeconomic backgrounds. Derived from the MultiYearRate subclass of the weak entity set Rate. The Rate ISA hierarchy is complete and joint, therefore there are no Rate tuples in our dataset, only either MultiYear or Yearly rates, and for each year (the key of Rate) there is both a Yearly and many MultiYearRates. Each tuple of MultiYearRates is supported by a School ID, starting year, and time period of the statistic. MutliYearRates have a timeperiod of either 4, 5, or 6 years. Tuples contain percentages of graduation for various divisions of student population (Hispanic, Economically Disadvantaged, etc.), 4, 5, or 6 years after entering secondary school.

- YearlyRates(schoolid, startyear, totalgrads, twofouryearcollegepercent, specializeddegreepercent, totalcollegeboundpercent, totalndgpsspercent, totalpostsecondarypercent, endyear)

This relation compares the gradutes going to universities, specialized college, etc. Derived from the YearlyRate subclass of the weak entity set Rate. For each year and school there is one YearlyRates tuple so there is no key within the entity. YearlyRates tuples hold yearly percentages of graduate's schooling plans post-secondary school (Nondegree-granting school, Specialized institution, etc.)

Constraints

Domain Constraints:

1. An attribute representing the length of time a statistic was tracked, 'timeperiod,' requires a domain constraint of 4, 5, or 6 representing the three lengths of time in years present in our data set.
2. Many attributes are percentages, requiring a 0-100 domain constraint:

- Male
- Female
- White
- Hispanic
- Black
- Asian
- Multiracial
- Americanindianalaskan
- Specialed
- Econdisadv
- Migrant
- Ell
- Twofouryearcollegepercent
- Specializeddegreepercent
- Totalcollegeboundpercent
- Totalndgpsspercent
- totalpostsecondarypercent

This is how the constraints are listed out in the code:

```
    CONSTRAINT aiaratescheck CHECK (((americanindianalaskan >= (0)::double precision) AND
(americanindianalaskan <= (100)::double $
    CONSTRAINT asianratescheck CHECK (((asian >= (0)::double precision) AND (asian <=
(100)::double precision))),
    CONSTRAINT blackratescheck CHECK (((black >= (0)::double precision) AND (black <=
(100)::double precision))),
    CONSTRAINT econdisadvratescheck CHECK (((econdisadv >= (0)::double precision) AND
(econdisadv <= (100)::double precision))),
    CONSTRAINT ellratescheck CHECK (((ell >= (0)::double precision) AND (ell <= (100)::double
precision))),
    CONSTRAINT femaleratescheck CHECK (((female >= (0)::double precision) AND (female <=
(100)::double precision))),
    CONSTRAINT hispanicratescheck CHECK (((hispanic >= (0)::double precision) AND (hispanic <=
(100)::double precision))),
    CONSTRAINT maleratescheck CHECK (((male >= (0)::double precision) AND (male <= (100)::double
precision))),
```

```
    CONSTRAINT migrantratescheck CHECK (((migrant >= (0)::double precision) AND (migrant <=
(100)::double precision))),
    CONSTRAINT multiracialratescheck CHECK (((multiracial >= (0)::double precision) AND (multiracial
<= (100)::double precision))),
    CONSTRAINT specialedratescheck CHECK (((specialed >= (0)::double precision) AND (specialed <=
(100)::double precision))),
    CONSTRAINT whiteratescheck CHECK (((white >= (0)::double precision) AND (white <=
(100)::double precision)))
CONSTRAINT specializeddegreeratescheck CHECK (((specializeddegreepercent >= (0)::double precision)
AND (specializeddegreeperce$
    CONSTRAINT totalcollegeboundratescheck CHECK (((totalcollegeboundpercent >= (0)::double
precision) AND (totalcollegeboundperce$
    CONSTRAINT totalndgpssratescheck CHECK (((totalndgpsspercent >= (0)::double precision) AND
(totalndgpsspercent <= (100)::doubl$
    CONSTRAINT totalpostsecondaryratescheck CHECK (((totalpostsecondarypercent >= (0)::double
precision) AND (totalpostsecondarype$
    CONSTRAINT twofouryearratescheck CHECK (((twofouryearcollegepercent >= (0)::double precision)
AND (twofouryearcollegepercent <$
```

Keys and Dependencies:

1. SchoolID is the primary key for the School relation. AUN is also a key however not used in the database as one. These two values are assigned by the state of Pennsylvania to uniquely identify school entities. Therefore, these two must be unique amongst all tuples in the school relation.
    - SchoolID and AUN are also foreign keys for Multiyear and Yearly rates; therefore for every rate tuple, its SchoolID must exist in the School relation.
2. The key for the Multiyear relation is SchoolID, Timeperiod, and StartYear, because for every year per school, there are 3 time periods for which that year's freshman cohort was tracked. For each unique combination of these three attributes there is one unique tuple of rates.
3. The key for Yearly is simply SchoolID and StartYear, because for each school there is only one set of data per year of yearly statistics, therefore for every unique combination of these two attributes there is exactly one tuple of yearly rates.

Loading In Data Procedure:

First, the various spreadsheets had to be cleaned and standardized. Glossary and intro pages, titles, blank rows/columns, and extraneous text at the bottoms of the data sheets were removed. Column headers were renamed to match psql attribute titles exactly. The resulting spreadsheets have a uniform format where the first value of the first tuple is in row 2, column 1.

Extraneous columns were then removed and the remaining reordered into a common format match that in psql. We had to remove all the % and , signs that were behind numbers because we wanted to represent percentages as floats and the % sign would have forced us to represent percentages as a string.

We created tables in our database that matched the columns in the data, most of these columns we had condensed in our multiyearrates, yearlyrates and school tables. However, the data contains those columns and we can't just delete those columns from the data.

To load in the files for the multiyearrates table:

squad=> create table tempcohort (schoolid integer, totalcohort varchar(20), totalgrads varvhar(20), male float, female float, white float, hispanic float, black float, asian float, multiracial float, americanindianalaskan float, specialed float, econdisadv float, migrant float, ell float, nativehawaiianpacificislander float);

This tempcohort table relates directly to the multiyearrates table. The only difference between the two is the timeperiod and startyear attribute in the multiyear table. These values are taken from the title of the document "2010-2011Cohort4year.csv" for example. We can't add these attributes to the tempcohort table because they aren't in the data file. This is why in the script below, after we copy over the data to the tempcohort table, we have to add timeperiod and startyear columns and manually insert the values. There are 2 inserts that we do for each file: the insert into schools is selective because we only want to add schools to the table that haven't already been added in prior files. Most of these schools were added in the first file but sometimes there are outliers in schools. The next insert line is actually adding in all the data (male grad rate, female grad rate...) from the tempcohort table to the multiyearrates table.The timeperiod and startyear columns are deleted once the data from tempcohort is transferred over to multiyearrates table because the next file won't have these columns. The data in the table is dropped for the next file of data.

This is the general script we used for every Cohort file to load data into the multiyearrates table:

squad=> \copy tempcohort from '/sh2/klererf/Downloads/20112012cohort4year.csv';

squad=> alter table tempcohort add column startyear integer;

squad=> alter table tempcohort add column timeperiod integer;

squad=> update tempcohort set startyear=2011;

```
squad=> update tempcohort set timeperiod=4;

squad=> insert into school(id, aun, name,type, lea) select schoolid, aun, name,type, lea from
tempcohort where schoolid not in(select id from school);

squad=> insert into multiyearrates (schoolid, startyear, timeperiod, totalcohort, totalgrads, male,
female, white, hispanic, black, asian, multiracial, americanindianalaskan, specialed, econdisadv,
migrant, ell, nativehawaiianpacificislander) select schoolid, startyear, timeperiod, totalcohort,
totalgrads, male, female, white, hispanic, black, asian, multiracial, americanindianalaskan,
specialed, econdisadv, migrant, ell, nativehawaiianpacificislander from tempcohort;

squad=> alter table tempcohort drop column timeperiod;
squad=> alter table tempcohort drop column startyear;
squad=> delete from tempcohort;
```

To load in the files for the yearlyrates table:

```
squad=> create table tempgrads (schoolid integer, totalgrads
varchar(20),twofourcollegeyearpercent float, specializeddegreepercent float,
totalcollegeboundpercent float, ndgpsspercent float, totalpostsecondaryboundpercent float);
```

This tempgrads table relates directly to the yearlyrates table. The only difference between the two is the startyear attribute in the yearlyrates table. These values are taken from the title of the document “Graduates201011.csv” for example. We can’t add these attributes to the tempgrads table because they aren’t in the data file. This is why in the script below, after we copy over the data to the tempgrads table, we have to add the startyear column and manually insert the values based on the file name. There are 2 inserts that we do for each file: the insert into schools is selective because we only want to add schools to the table that haven’t already been added in prior files. Most of these schools were added in the first file but sometimes there are outliers in schools. The next insert line is actually adding in all the data (2-4 year college percent, specialized degree percent...) from the tempgradstable to the yearlyrates table.The startyear column is deleted once the data from tempgrads is transferred over to yearlyrates table because the next file won’t have these columns. The data in the table is dropped for the next file of data.

```
squad=> \copy tempgrads from '/sh2/klererf/Downloads/Graduates201011.csv';
squad=> alter table tempgrads add column startyear integer;
squad=> update tempgrads set startyear=2010;
```

```
squad=> insert into school (id, aun, name, county, lea) select schoolid, aun, name, county, lea
from tempgrads where schoolid not in (select id from school);

squad=> insert into yearlyrates (schoolid, startyear, totalgrads, twofouryearcollegepercent,
specializeddegreepercent, totalcollegeboundpercent, totalndgpsspercent,
totalpostsecondarypercent) select schoolid, startyear, totalgrads,twofourcollegeyearpercent,
specializeddegreepercent, totalcollegeboundpercent, ndgpsspercent,
totalpostsecondaryboundpercent from tempgrads;

squad=> alter table tempgrads drop column startyear;
squad=> delete from tempgrads;
```

Future Data Uploads:

Future data uploads will likely require a process similar to the one we went through to upload the current data. The data from previous years was “dirty” and inconsistent, making the process of using a single repeatable script difficult. Unfortunately, future data will probably have the same inconsistencies of past years. The first step to adding additional data would be to confirm that the attributes of the data match what we currently have. Next, as with the data from past years, erroneous symbols and punctuation would have to be removed from the data. Once data cleanliness is established, the data can them be loaded the same way that past years have been. Attributes must be ordered in the correct way, as shown in the section below. If correct order is not established data will not load correctly. The script will load the raw data into a temporary table which will then be moved into the multiyearrates and yearlyrates tables. The downside with this process is although effective, it is not as straightforward for an inexperienced user as it is for our group. Someone who has limited or no experience using databases or SQL would likely have a difficult time with an upload.

Correct Attribute Order

Cohort Data Temp Table Attributes (leatype, aun, lea, schoolid, name, type, totalgrads, totalcohorts, male, female, white, hispanic, black, americanindianalaskan, multiracial, nativehawaiianpacificislander, asian, econdisadv, migrant, ell, specialed)

Graduates by Public school (county, aun, lea, schoolid, name, totalgrads, totalcollegebound, totalcollegeboundpercent, twofouryearcollege, twofourcollegeyearpercent, totalpostsecondarybound, totalpostsecondaryboundpercent, ndgpss, ndgpsspercent, specializedegree, specializeddegreepercent)

Citations:

"Graduate Data and Statistics." *Pa.gov*. PA Department of Education, 2016. Web. 23 Oct. 2016.