```
In [ ]: %%sql

/*
The entities we decided to include in our schema are as follows: institutions, students, financials,
and academics. Each of these entities are linked to a single college, so there is a bit of inherent
redundancy in the primary keys used. However, we believe that this split is beneficial as each entity
relates to a unique part of each college. For example, if a user is interested in the financial
information regarding colleges in a specific region, they may not be interested in aspects such as
the number of students at that university. It is entirely possible to contain every piece of
information about these schools in a single table, but this would be cluttered and ineffective for
future analysis.
*/

DROP TABLE IF EXISTS Institutions CASCADE;
DROP TABLE IF EXISTS Students CASCADE;
DROP TABLE IF EXISTS Financials CASCADE;
DROP TABLE IF EXISTS Academics CASCADE;

/*
Creates a table for the institutions, here each record will correspond to an institution or a college.
The purpose of this table is to have a masters table that stores all the information about the colleges
that will not change very often. Here, if there is a change in some basic college information, we will
just update the table and not store the history. This is to make sure we have a table that is smaller,
easier to read, and more efficient to join to others(will be a one to many join with all other tables
on institution_id)
*/

CREATE TABLE Institutions (
   institution_id SERIAL PRIMARY KEY,
   name TEXT NOT NULL,
   accredagency TEXT, -accrediting agency
   control INT CHECK (control IN (1, 2, 3)),
   region INT CHECK (region BETWEEN 0 AND 9),
   CCbasic INT CHECK (CCbasic <= 33) --- Carnegie classifier
);


/*
Creates a table that keeps track of information about a university's students in a given year.
Information here would be a students' demographics, personal information (all are averages here across
```

```sql
all students except number of students) before or after graduation. Here, the primary key will be a
combination of both institution and year, and it will have a foreign key to the institutions table to
get further information about the institution. This will also have students' financial information in
aggregate because the point of the student table is to keep track of what the student body looks like
(which includes their financial status)
*/

CREATE TABLE Students (
   institution_id INT REFERENCES Institutions(institution_id),
   year INT CHECK (year > 0 AND year <= EXTRACT(YEAR FROM CURRENT_DATE)),
   adm_rate FLOAT CHECK (adm_rate >= 0 and adm_rate <= 1), --- Admission rate
   num_students INT CHECK (num_students >= 0),
   act FLOAT CHECK (act >= 1 and act <= 36), --- ACT exam score
   cdr2 FLOAT CHECK (cdr2 >= 0 and cdr2 <= 1), --- 2 year default rate of student loans
   cdr3 FLOAT CHECK (cdr3 >= 0 and cdr3 <= 1), --- 3 year default rate of student loans
   first_gen FLOAT CHECK (first_gen >= 0 and first_gen <= 1), --- percentage of first-gen students
   avg_family_income INT CHECK (avg_family_income >= 0),
   PRIMARY KEY (institution_id, year)
);



/*
This will create a table about the status of the costs about the university itself.
These are dictated or received by the university as a whole as opposed to the information
in the students table being a gauge of what the student population looks like.
Here, we will see metrics such as how much the university is charging for certain programs or
how much scholarship they are giving by policy.
This also has a primary key of institution and year, showing what the university is doing in
terms of financial information for a year. It also has a foreign key referencing institutions,
since we will want to connect it to basic information about the university as a whole.
*/

CREATE TABLE Financials (
   institution_id INT REFERENCES Institutions(institution_id),
   year INT CHECK (year > 0 AND year <= EXTRACT(YEAR FROM CURRENT_DATE)),
   tuitionfee_in INT CHECK (tuitionfee_in >= 0), --- In state tuition
   tuitionfee_out INT CHECK (tuitionfee_out >= 0), --- Out of state tuition
   tuitionfee_prog INT CHECK (tuitionfee_prog >= 0), --- Average tuition for program-year -institutions
   tuitfte INT CHECK (tuitfte >= 0), --- Net tuition revenue per student
   avgfacsal INT CHECK (avgfacsal >= 0), --- Average faculty salary
   PRIMARY KEY (institution_id, year)
);
```

```sql
/*
The academics table will store information about the universities’ metrics related to their classrooms.
This will include information about what degrees they offer, what type of degrees they offer, and
other metrics about what goes on in their class.
Again, this is in aggregate for a whole year as the primary key here is institution and year and we
will reference the institutions table as the foreign key.
*/

CREATE TABLE Academics (
   institution_id INT REFERENCES Institutions(institution_id),
   year INT CHECK (year > 0 AND year <= EXTRACT(YEAR FROM CURRENT_DATE)),
   preddeg TEXT CHECK (preddeg BETWEEN 0 AND 4), --- predominant degree awarded
   highdeg INT CHECK (highdeg BETWEEN 0 AND 4), --- highest degree awarded
   stufacr FLOAT CHECK (stufacr >= 0 AND stufacr <= 1), --- student to faculty ratio
   PRIMARY KEY (institution_id, year)
);
```