Phase 2: Database

Sierra Allred

July 25, 2016

The second phase of my thesis was the creation of a MySQL database to organize data collected from College Scorecard and AreaVibes.com. Because the visualization portion of my project was done with a team before I had database experience, the visualization and database are not connected at this time. The next step of my project would be to use Node.js to connect the visualization to the database with the queries I have created.

This document will describe how I collected the data I needed from AreaVibes, how the project database is organized, and the queries that I created to collect data for the visualization and statistical analysis portions of my project.

1 Collection of AreaVibes Data

For the statistical a analysis portion of my project I needed cost of living (COL) data for various cities throughout the United States. AreaVibes.com gives each city a cost of living rating, but their data was not readily available from an API, and was spread throughout the site on various web pages. I used Scrapy, a Python based web crawling language, to collect the data I needed in an organized csv file.

I started by creating a Scrapy item that defined all the fields I wanted to collect. The COLarea was the COL of a certain area of a city, and COLcity was the overall COL of the city. I also collected the City, Area, and State.

```
from scrapy.item import Item, Field

class ColItem(Item):
    # define the fields for item:
    COLarea= Field()
    COLcity= Field()
    City = Field()
    State = Field()
    Area = Field()
```

Next I created the spider I would use to crawl the website. I went to AreaVibes robots.txt file to find the sitemap, and use that to navigate to a xml file containing all the links that held data on each city and its COL. I used that xml as the starting url for my spider to crawl. My spider collects the page from each url, and then creates an item to store the data it collects. I used html tags to direct the spider to the data in the page, which it then extracts and stores.

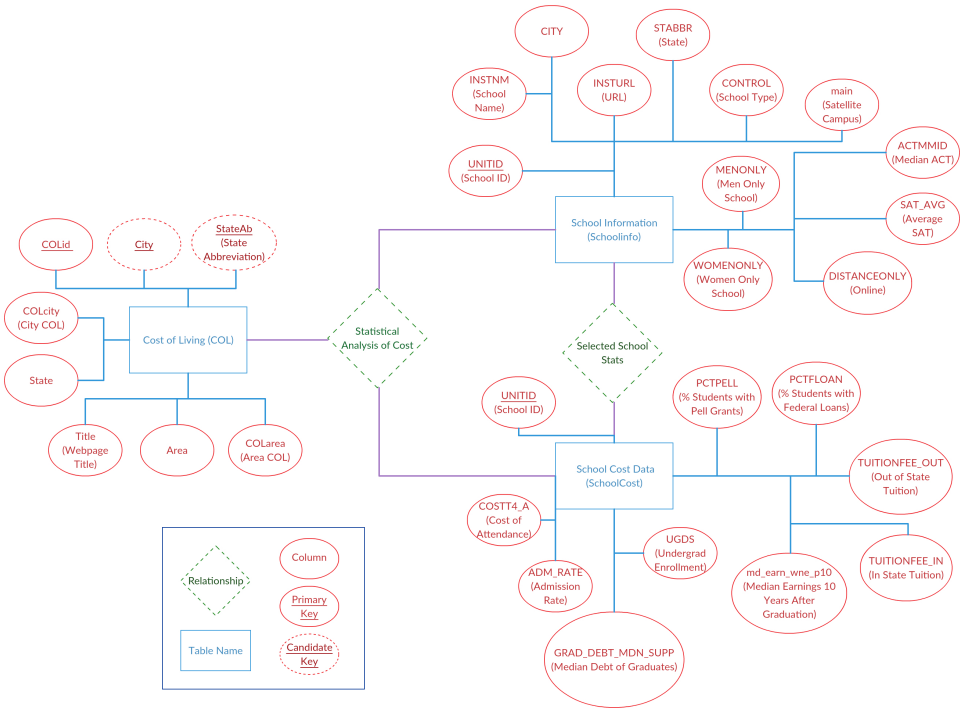
```
import scrapy
from scrapy.spiders import SitemapSpider
from COL.items import ColItem

class COLSpider(SitemapSpider):
    name = "COLspidey"
    sitemap_urls = ['http://www.areavibes.com/sitemap/sm_hood_COL.xml']

    def parse(self, response):
       page = response.xpath("//html")
       for sel in page:
           item = ColItem()
           item["title"] = sel.xpath("//title/text()").extract()
           item["COLarea"] = sel.xpath("//tr[@class='summary major']/td[2]/text()").extract()
           item["COLcity"] = sel.xpath("//tr[@class='summary major']/td[3]/text()").extract()
           item["City"] = sel.xpath("//nav[@class='breadcrumbs']/a[3]/text()").extract()
           item["State"] = sel.xpath("//nav[@class='breadcrumbs']/a[2]/text()").extract()
           item["Area"] = sel.xpath("//nav[@class='breadcrumbs']/a[4]/text()").extract()

           yield item
```

I used the command line to crawl my spider and store all the data in an organized csv file which could easily be added to the project database.

2 Database Organization

The project database is a very simple MySQL database made up of 3 tables. One table stores all of the cost related data about each university in the United States, and another holds all other relevant information about each school such as the location, url, name, and test scores. The last table holds all of the COL data collected from AreaVibes.

Below is the entity-relationship model for the project database outlining the tables, columns, and keys and their relationships.



3 Queries

I have created all the queries needed for the visualization and statistical analysis portions of my project, this section will describe each of these.

The create table statement used for the COL table.

```
CREATE TABLE `scorecard`.`COL` (
 `COLid` INT NOT NULL,
 `City` VARCHAR(45) NULL,
 `COLcity` INT NULL,
 `COLarea` INT NULL,
 `StateAb` VARCHAR(45) NULL,
 `Title` VARCHAR(45) NULL,
 `Area` VARCHAR(45) NULL,
 `State` VARCHAR(45) NULL,
 PRIMARY KEY (`idCOL`),
 UNIQUE('City', 'State'));
```

This statement was used to load the COL data in the csv into the COL table.

```
load data local infile '/home/sierra/Desktop/COLdata.csv'
into table COL fields terminated by ','
enclosed by '"'
lines terminated by '\n'
IGNORE 1 LINES
(idCOL, City, COLcity, COLarea, StateAb, title, Area, State);
```

This query was used to collect the cost data I needed for statistical analysis. I had to join all 3 tables by School ID, City, and State. Because so much data was missing from the College Scorecard dataset, I specified that the query only return rows that did not have NULL or PrivacySuppressed values.

```
select DISTINCT
  cols.COLcity COL,
  sc.COSTT4_A cost,
  sc.md_earn_wne_p10 earnings,
  sc.GRAD_DEBT_MDN_SUPP debt
from
  SchoolCost sc,
  COLs cols,
  Schoolinfo si
where
  si.STABBR = cols.StateAb
  and si.CITY = cols.City
  and si.UNITID = sc.UNITID
  and sc.COSTT4_A not like 'NULL'
  and sc.md_earn_wne_p10 not like 'NULL'
  and sc.md_earn_wne_p10 not like 'Priva'
  and sc.GRAD_DEBT_MDN_SUPP not like 'NULL'
  and sc.GRAD_DEBT_MDN_SUPP not like 'PrivacySuppressed';
```

These queries create temporary tables. schoolList holds the name and id of all the schools returned by the user driven filter. All the variables contained by [] are placeholders for values returned by the filter. graph selects all the data needed for the different options of the bar chart in the visualization from schoolList.

```
where
schoolList as (
select
  si.UNITID sid,
  si.INSTNM sname
from
  SchoolCost sc,
  SchoolInfo si
where
  sc.UNITID = si.UNITID
  and si.SAT_AVG < [satHi]
  and si.SAT_AVG > [satLo]
  and si.ACTCMMID < [actHi]
  and si.ACTCMMID > [actLo]
  and sc.TUITIONFEE_OUT < [tutHi]
  and sc.TUITIONFEE_OUT > [tutLo]
  and sc.TUITIONFEE_IN < [tutHi]
  and sc.TUITIONFEE_IN > [tutLo]
  and sc.UGDS < [sizeHi]
  and sc.UGDS > [sizeLo]
  and si.CONTROL = [type]
),
graph as (
select
  dat.ADM_RATE admin,
  dat.COSTT4_A cost,
  dat.md_earn_wne_p10 earnings,
  dat.GRAD_DEBT_MDN_SUPP debt
from
  SchoolCost sc,
  SchoolList sl
where
  sl.UNITID = sc.UNITID
)
```

This query is for the part of the visualization that gives you information about an individual school selected from the list of filtered schools. [selectedSchool] is a placeholder for ID of the school selected from that list.

```
select
  si.INSTNM name,
  si.CITY city,
  si.STABBR state,
  si.INSTURL url,
  si.MENONLY men,
  si.WOMENONLY women,
```

```
  si.main satellite,
  si.DISTANCEONLY onlineOnly,
  sc.ADM_RATE admin,
  sc.UGDS undergradEnroll,
  si.ACTCMMID medACT,
  si.SAT_AVG avgACT,
  sc.TUITIONFEE_IN inTuition,
  sc.TUITIONFEE_OUT outTuition,
  sc.COSTT4_A cost,
  sc.PCTPELL pell,
  sc.PCTFLOAN loan,
  sc.GRAD_DEBT_MDN_SUPP debt,
  sc.md_earn_wne_p10 earnings
from
  SchoolCost sc,
  SchoolInfo si
where
  si.UNITID = [selectSchool]
  and si.UNITID = sc.UNITID;
```

This query provides the data needed for the selected schools list to be ordered by. [selectOrder] is a placeholder for the option selected to order the list.

```
select
  si.UNITID sid,
  si.INSTNM name,
  sc.UGDS undergradEnroll,
  sc.GRAD_DEBT_MDN_SUPP debt,
  sc.COSTT4_A cost,
  sc.ADM_RATE admin,
  sc.md_earn_wne_p10 earnings,
from
  SchoolCost sc,
  SchoolInfo si,
  SchoolList sl
where
  sc.UNITID = sl.UNITID
  and sc.UNITID = si.UNITID
order by
  sc.[selectOrder];
```