ALUMNI RELATIONSHIP MANAGEMENT SYSTEM

Group No.: Group 14

Student Names: Gaurav Tople and Sai Vineeth

Executive Summary:

The purpose of this project is to establish the connection between alumni and current students in the institution. The key functions that we want to include this project are Referral/postings, Events for alumni and Discussion forum. According to the above scenario we first developed Entity relationship model which successfully integrated the above-mentioned functions into relations between students, alumni and departments of the institution. With the help of UML, we generalized the relations and transformed them in Normalized relational Model. Through the develop model we used the data extensively and were able to develop useful insights in python. The implementation of NoSQL using collections in MongoDB helped us to retrieve some useful insights on Job postings and Referral dashboard. This model helps the alumni to easily post about the job offering where it reaches the desired set of students in the institution as the data is well designed throughout the tables with necessary attributes. Thus, students can easily contact the alumni for suggestions and referrals in their desired field of interest.

I. Introduction

1. Problem

We all came from different backgrounds and different societies to a school. As international student, it's always difficult to cope without an advisor or any certain directed path. To get a directed path it's very important to have relation with the alumni of the institute. To connect the alumni with the college and bridge the gap between the current students, it's essential to have a system that can track the alumni, current students and institution's data and events. Schooling is a big part in one's life and having connected to your school at any time in your life is what everyone wanted.

The Alumni management system focuses on the association of alumni (graduates) or, broadly, of former students to the college and their present students is a key bridge to any institution. This association can open the key door to often organize social events; publish newsletters or magazines raise funds for the organization. These events provide a variety of benefits and services that help alumni maintain connections to their educational institutions and fellow graduates.

2. Goal

Our goal is to bind the alumni and students to the institution by developing an effective Alumni Management System where it opens a new opportunity to post referrals, organize events, initiate discussions.

The model will be developed for a school to have a record of its alumni and a functionality to connect them with their current students for guidance and reference. And it should support following functionalities:

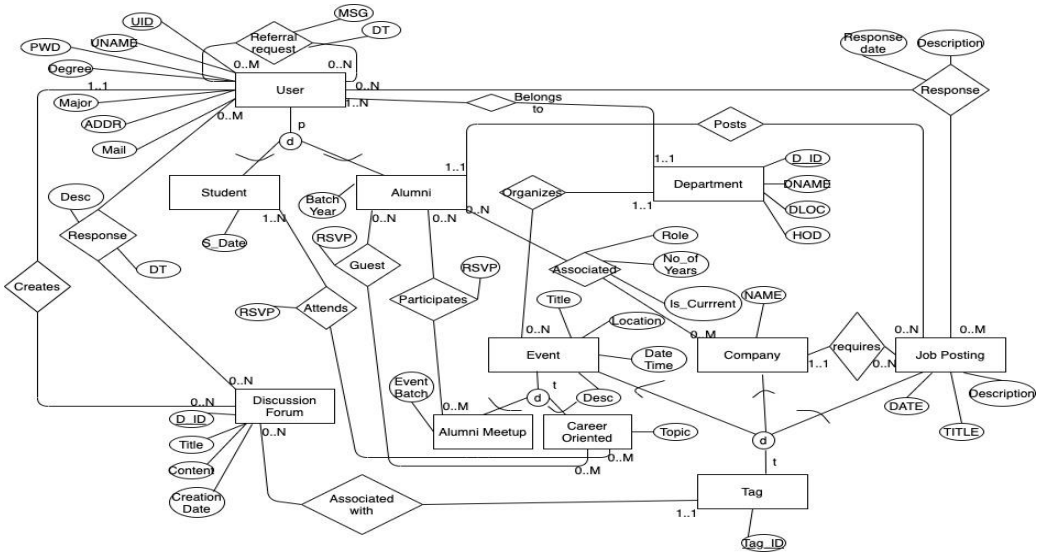
a) **Referrals/Postings:** Current students, alumni can request for referrals from alumni who are working or associated with the respective company through the website. Alumni students can post the job offerings, which will be available to all the other members on the website to look for. For each posting, there will be an associated alumnus who will be responding to the questions from the other members of the site regarding the post.
b) **Alumni Events:** Meetups can be organized/initiated by the college department for certain alumni batch(year), and the same can be finalized through the RSVP from alumni of that respective batch. The notifications should be sent to the batch respectively. College can also invite alumni for special events organized for current students (For Example: Tech Talks, Career guidance events, success stories, etc.)
c) **Discussion Forum:** Members of the group can start a discussion forum pertaining to college, company, Research paper, and the alumni, students related to those in one or other way will be able to join the forum. A notification mail should be sent to the members who are aligned with the intent of the question.

3. Requirements:

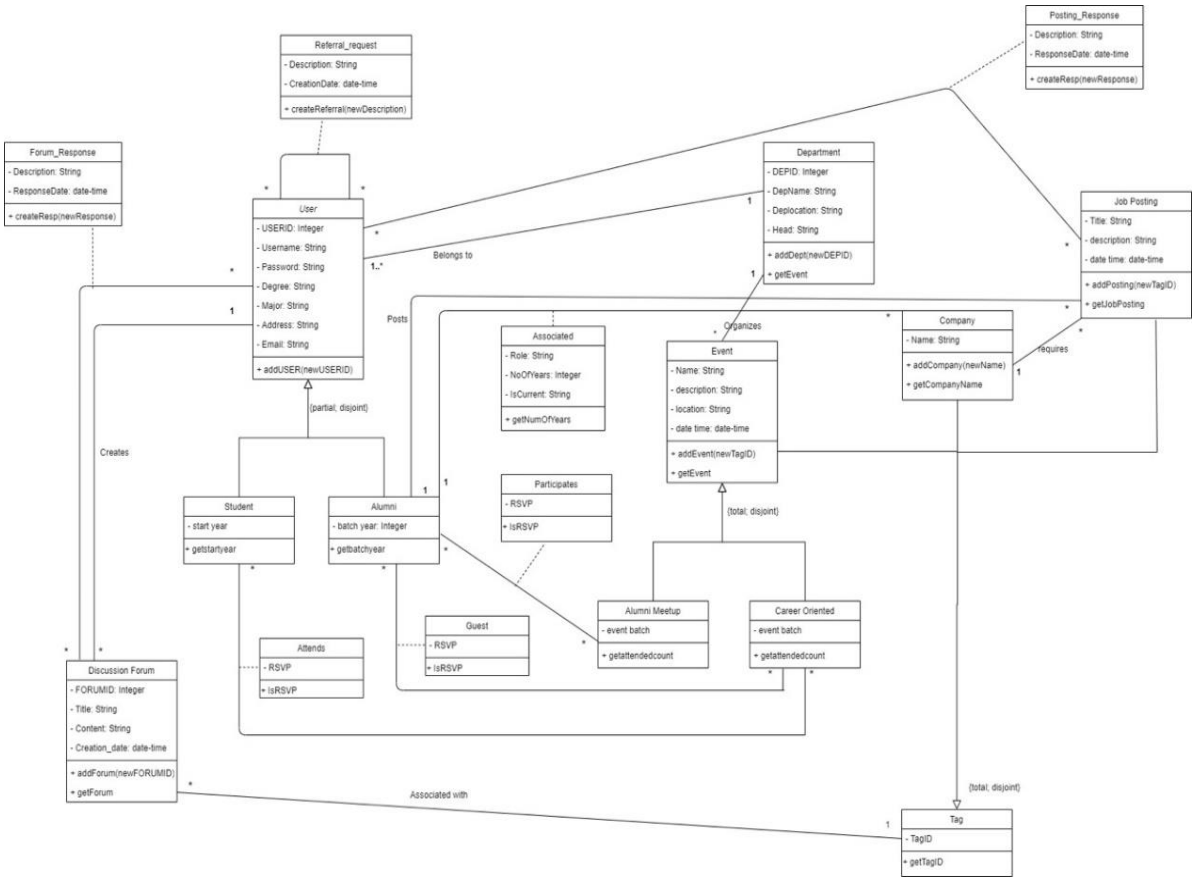
To draw EER and UML class diagram we used draw.io tool. The SQL database used was MySQL. We performed NoSQL implementation using MongoDB. Spyder: the python IDE is used to demonstrate the access and data retrieval of MySQL database using Python. MongoDB Compass and virtual playground is used to implement MongoDB NoSQL database.

II. Conceptual Data Modeling

We have developed EER model with domains and worked on cardinalities between them, as you can see from the below Entity Relation Model:



Like EER diagram, the UML class diagram is another diagrammatic representation which shows abstractions and cardinalities between the data.



Some Constraints that we were not able to show on EER and UML:

- We can't show the limit on number of students, alumni assigned to a department.
- The limit on Event invites.
- The limit on number of referrals that an alumnus can post.
- The time that a discussion can last.

III. Mapping Conceptual Model to Relational Model

The EER and UML representation of Alumni Management System has been successfully converted into relational model with all the domains, attributes, constraints (Relational Constraints), relations in 4NF.

Relational Model Representation:

Tag(Tag_ID)

Department(Dep_id, Dep_name, Dep_location, Head)

User(User_id, username, password, degree, major, address, email, *Dep_id*)

Discussion_forum(Forum_ID, title, content, creation_date, *tag_id, user_id*)

Student(*S_User_id*, Start_date)

Alumni(*A_User_id*, Batch_Year)

Event(*E_Tag_ID*, title, location, date_time, *Dep_ID*)

Alumni_Meetup(*A_E_Tag_ID*, batch_year)

Career_Oriented(*C_E_Tag_ID*, topic)

Company(*C_Tag_ID*, Name)

Job_posting(*J_Tag_ID*, title, description, date, *C_Tag_ID*, *A_User_ID*)

Posting_Response(*J_Tag_ID*, *User_id*, description, reponse_date)

Company_Association(*A_User_id*, *C_Tag_ID*, role, no_of_years, is_current)

Student_Attends(*S_User_id*, *C_E_Tag_ID*, RSVP)

Alumni_Guest(*A_User_id*, *C_E_Tag_ID*, RSVP)

Alumni_Participates(*A_User_id*, *A_E_Tag_ID*, RSVP)

Forum_Response(*Forum_ID*, *User_ID*, description, response_date)

Referral_request(*Sender_User_ID*, *Receiver_User_ID*, Description, Creation_Date)

The foreign keys(Referential Constraints) are italized and primary keys are underlined.

IV Implementation of Relation Model via MySQL and NoSQL

Using the developed relational mode, we have created the Alumni Management Database with all the tables listed above with the constraints followed by it. We have created multiple triggers to align with the Tag ID generation across the tables.

Triggers

All the events below are aligned with the autogenerated Tag_ID across the events and event associated tables

```
use arms;
DROP TRIGGER if exists COMPANY_TAG_INSERT
DELIMITER //
CREATE TRIGGER COMPANY_TAG_INSERT
AFTER INSERT
ON company FOR EACH ROW
BEGIN
        INSERT INTO TAG VALUES(NEW.c_tag_id);
END; //
DELIMITER ;

DROP TRIGGER if exists JOB_POSTING_TAG_INSERT
DELIMITER //
CREATE TRIGGER JOB_POSTING_TAG_INSERT
AFTER INSERT
ON JOB_POSTING FOR EACH ROW
BEGIN
        INSERT INTO TAG VALUES(NEW.j_tag_id);
END; //
DELIMITER ;

use arms
DROP TRIGGER if exists EVENT_TAG_INSERT
DELIMITER //
CREATE TRIGGER EVENT_TAG_INSERT
AFTER INSERT
ON EVENT FOR EACH ROW
BEGIN
        INSERT INTO TAG VALUES(NEW.e_tag_id);
END; //
DELIMITER ;
```

Stored procedure

Some of the stored procedures which are useful while inserting into the posting tables as if when required are listed below:

```
use arms;
DROP PROCEDURE IF EXISTS `INSERT_INTO_COMPANY`;
DELIMITER //
CREATE PROCEDURE `INSERT_INTO_COMPANY`(IN cname VARCHAR(255))
BEGIN
   DECLARE NEW_TAG_ID INT(11);
   SELECT MAX(tag_id) + 1 INTO NEW_TAG_ID FROM TAG;
   INSERT INTO COMPANY (C_TAG_ID, name) VALUES(NEW_TAG_ID, cname);
END; //
DELIMITER ;
```

```
use arms;
DROP PROCEDURE IF EXISTS `CREATE_EVENT`;
DELIMITER //
CREATE PROCEDURE `CREATE_EVENT`(IN TITLE VARCHAR(255), DEP_ID
INT(11), LOCATION VARCHAR(255), DATE_TIME DATE)
BEGIN
   DECLARE NEW_TAG_ID INT(11);
   SELECT MAX(tag_id) + 1 INTO NEW_TAG_ID FROM TAG;
   INSERT INTO EVENT (E_TAG_ID, title, Dep_ID, Location, Date_time)
VALUES(NEW_TAG_ID, TITLE, DEP_ID, LOCATION, DATE_TIME);
END; //
DELIMITER ;
```

```
use arms;
DROP PROCEDURE IF EXISTS `CREATE_JOB_POSTING`;
DELIMITER //
CREATE PROCEDURE `CREATE_JOB_POSTING`(IN C_TAG_ID INT(11),
A_USER_ID INT(11), TITLE VARCHAR(255), DESCRIPTION VARCHAR(255),
DATE DATE)
BEGIN
   DECLARE NEW_TAG_ID INT(11);
   SELECT MAX(tag_id) + 1 INTO NEW_TAG_ID FROM TAG;
   INSERT INTO JOB_POSTING (J_TAG_ID, C_TAG_ID, A_USER_ID, title,
description, date) VALUES(NEW_TAG_ID, C_TAG_ID, A_USER_ID, TITLE,
DESCRIPTION, DATE);
END; //
DELIMITER ;
```

These are some of the complex Queries connected through multiple queries helps to get some important data:

Scenario 1: Topic of career-oriented events held in 2018 along with number of students who attended that event i.e number of students who RSVPed 'Yes(1)'

```
select CO.topic,count(s_user_id) AS Num_of_Students_Attending
from event E, career_oriented CO, student_attends SA
where E.e_tag_id=co.c_e_tag_id
and CO.c_e_tag_id = SA.c_e_tag_id
and year(E.Date_time)=2018
and SA.rsvp = 1
group by CO.topic;
```

topic	Num_of_Students_Attending
Alumni Interact - Construction Management	18
Alumni Interact - Data Analytics	9
Alumni Interact - Telecommunication	10
Entrepreneurship	9
Full Time Search Panel 1	17
Full Time Search Panel 2	21
Industry Exposure	9
Job Search	29
Personal Development	7
Programming skills	9

Scenario 2: Name and batch of student who attended a career oriented event called 'Job Posting'. Batch should be determined from start date i.e if start date is '2019-08-20' then the batch name is 'Fall 2019'.

```
select U.username as Name,
        CASE
   WHEN MONTH(S.START_DATE) = 8 THEN CONCAT('FALL
',YEAR(S.START_DATE))
   ELSE CONCAT('SPRING ',YEAR(S.START_DATE))
   END AS Batch
from career_oriented CO, student_attends SA, student S, user U
where CO.C_E_TAG_ID = SA.C_E_TAG_ID
and SA.S_User_id = S.S_user_id
and S.S_user_id = U.User_id
and SA.rsvp = 1
and CO.Topic='Job Search';
```

Name	Batch
Phil	FALL 2016
Gail	FALL 2016
Kippie	FALL 2016
Marcellus	FALL 2016
Gram	FALL 2016
Gillan	FALL 2016
Maritsa	FALL 2016
Camille	FALL 2016
Augie	FALL 2016

Scenario 3: Name, batch year and Number of Postings of the alumni user who has highest number of job postings.

```
select U.Username As Name,
A.Batch_Year,
count(J_Tag_ID) AS No_Of_Postings
from job_posting JP, Alumni A, User U
where JP.A_USER_ID = A.A_USER_ID
and A.A_USER_ID = U.User_ID
group by U.Username, A.Batch_Year
order by 3 desc  LIMIT 1;
```

	Name	Batch_Year	No_Of_Postings
	Marietta	2012	16

Scenario 4: Retrieve Name, Batch Year, Role and No_Of_years of association of the alumni users who are currently associated with the company named 'Infosys'.

```
select U.UserName as Name, A.Batch_Year, CA.role, CA.no_of_years
from company_association CA, Company C, Alumni A, User U
where CA.C_Tag_ID = C.C_Tag_ID
AND CA.A_User_id = A.A_User_ID
AND A.A_User_ID = U.User_ID
AND CA.is_current = 1
AND C.NAME = 'INFOSYS';
```

Name	Batch_Year	role	no_of_years
Reggi	2008	Engineer	2
Carin	2008	Analyst	3
Georgie	2010	Sr Engineer	2
Janie	2011	Developer	2
Iorgo	2013	Sr Engineer	2

A View is created to retrieve the Career Event data directly in python to perform visualization and Analytics on it.

```
CREATE VIEW TOP_CAREER_EVENTS (TOPIC, NumOfStudentsAttended) AS
select CO.Topic, Count(S_USer_ID) As NumOfStudentsAttended from career_oriented
CO, student_attends SA
where CO.C_E_Tag_ID = SA.C_E_Tag_ID
AND SA.rsvp=1
group by CO.topic
order by 2 DESC
LIMIT 5;
```

In the view of extension to NOSQL, we developed collections related to User and events. The structure of a document looks like below:


```
{
    "_id": {
        "$oid": "5e9bca0a16a83e49904652df"
    },
    "USER_ID": {...},
    "Name": "Talbert",
    "Department": "Physics",
    "Company": "Honeywell",
    "Degree": "Masters",
    "Major": "Computers",
    "Address": "31 Stoughton Crossing",
    "Email": "tmadison1@desdev.cn",
    "Alumni": {...},
    "Job opening postings": {...}
}
```

```
{
    "_id": {
        "$oid": "5e9bca4f16a83e49904652e1"
    },
    "USER_ID": {...},
    "Name": "Cecilio",
    "Department": "Robotics",
    "Company": "Microsoft",
    "Degree": "Bachelors",
    "Major": "Data Science",
    "Address": "2461 Buell Parkway",
    "Email": "ctompion2@loc.gov",
    "Alumni": {...},
    "Job opening postings": {...}
}
```


Scenario 1: To retrieve average Job postings available per department.

```
db.Alumniusers.mapReduce(
function() {
   emit(this.Department, this.Jobopeningpostings); },
   function(key, values) {return Array.avg(values)},
   {out:"Avg" }).find()
```

Scenario 2: To retrieve total Job postings available per Company.

```
db.Alumniusers.mapReduce(
   function() { emit(this.Company,this.Jobopeningpostings);
   },
   function(key, values) {return Array.sum(values)},
   {    out:"Count_total" }).find()
```

Aggregate functions **connection string in Java**

Scenario 3: To retrieve the count of alumni recruited by each company.

```
db.alumniusers.aggregate[{

   $group: {

      _id: '$Company',

      Count: {

         $sum: '$Alumni'
```

```
    }
  }
}, {  $sort: { Count: -1}}])
```

Scenario 4: To retrieve event's attendance per Department

```
db.Alumniusers.aggregate[
$group: { id: '$Department',
Total: {
    $sum: '$RSVP'
}}}, {
$sort: { Count: -1}}]
```

V. Database Access via R or Python

The Demonstration of the access to MySQL database using Python is shown below:

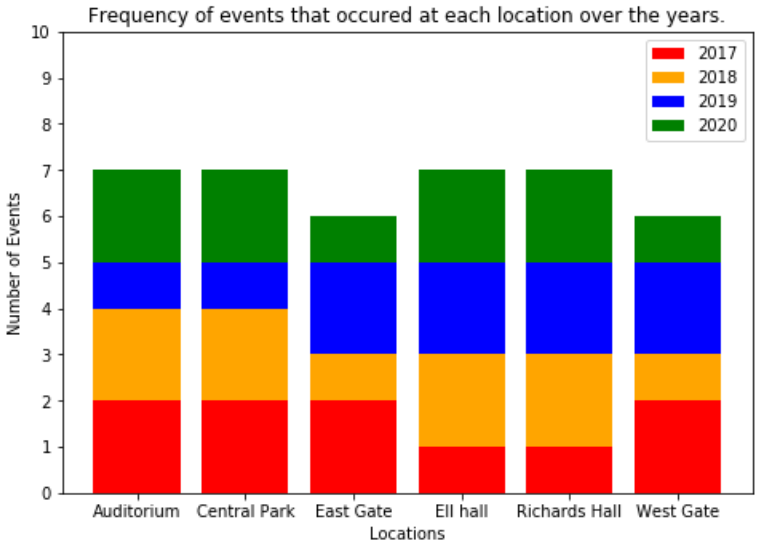
```
#example of python connecting to MySQL server and databases
#
import mysql.connector
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import squarify
#
from mysql.connector import Error
#
def connecttomysql(db):
    try:
        connection = mysql.connector.connect(host='127.0.0.1',
                                             database = db,
                                             user='root',
                                             password='',
                                             auth_plugin = 'mysql_native_password')


        return connection
    except Error as e:
        print("Error while connecting to MySQL", e)
        return e


def closeconnection(connection, cursor):
    if (connection.is_connected()):
        cursor.close()
        connection.close()
        print("MySQL connection is closed")

connection = connecttomysql('arms')
if connection.is_connected():
    db_Info = connection.get_server_info()
    print("Connected to MySQL Server version ", db_Info)
    cursor = connection.cursor()
```

```
    cursor.execute("select database();")
    record = cursor.fetchone()
    print("Your connected to database: ", record[0])
#
    sql_select_Query1 = '''SELECT location, YEAR(Date_time), count(*) AS NUM_OF_EVENTS
                        FROM EVENT e group by location, YEAR(Date_time);'''
    cursor = connection.cursor()
    cursor.execute(sql_select_Query1)
    records = cursor.fetchall()
    print(":\n")
    event_info = pd.DataFrame(records, columns = ['Location', 'Year', 'Number_Of_Events'])

    print("Frequency of events that occured at each location over the years.")
    stack1 = np.add(event_info[event_info.Year == 2017].Number_Of_Events.values,
                    event_info[event_info.Year == 2018].Number_Of_Events.values).tolist()
    stack2 = np.add(stack1, event_info[event_info.Year == 2019].Number_Of_Events.values)

    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    ax.bar(pd.unique(event_info.Location), event_info[event_info.Year == 2017].Number_Of_Events.values, color = 'r')
    ax.bar(pd.unique(event_info.Location), event_info[event_info.Year == 2018].Number_Of_Events.values,
           bottom = event_info[event_info.Year == 2017].Number_Of_Events.values, color='orange')
    ax.bar(pd.unique(event_info.Location), event_info[event_info.Year == 2019].Number_Of_Events.values,
           bottom = stack1, color='b')
    ax.bar(pd.unique(event_info.Location), event_info[event_info.Year == 2020].Number_Of_Events.values,
           bottom = stack2, color='g')
    ax.set_yticks(np.arange(0, 11, 1))
    ax.set_ylabel('Number of Events')
    ax.set_xlabel('Locations')
    ax.set_title('Frequency of events that occured at each location over the years.')
    ax.legend(labels=pd.unique(event_info.Year))
    plt.show()
```

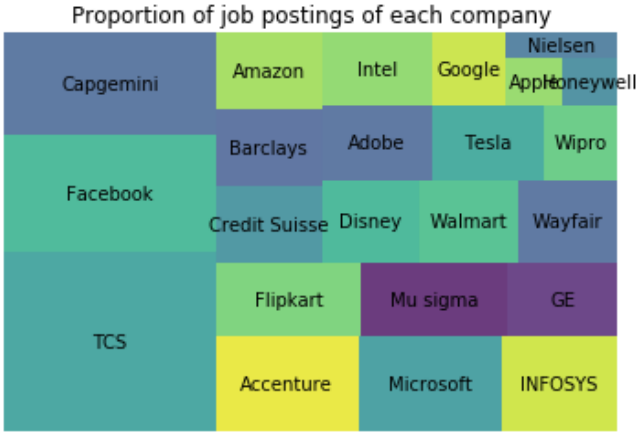
```
    sql_select_Query2 = '''select C.Name, Count(*) AS Num_Of_Postings from job_posting JP, Company C
                        where JP.C_Tag_ID = C.C_Tag_ID
                        group by C.Name
                        order by 2 Desc;'''
    cursor = connection.cursor()
    cursor.execute(sql_select_Query2)
    records = cursor.fetchall()
    print("\n")
    print("Proportion of job postings of each company")
    company_postings = pd.DataFrame(records, columns = ['CompanyName', 'NoOfPostings'])
    squarify.plot(sizes=company_postings['NoOfPostings'], label=company_postings['CompanyName'], alpha=.8 )
    plt.axis('off')
    plt.title('Proportion of job postings of each company')
    plt.show()

    #View Created in Database has been Called here:
    sql_select_Query3 = "select * from top_career_events LIMIT 5;"
    cursor = connection.cursor()
    cursor.execute(sql_select_Query3)
    records = cursor.fetchall()
    print("\n")
    top_career_event = pd.DataFrame(records, columns = ['Topic', 'NumOfStudentsAttended'])
    df = pd.DataFrame({'Students Attended': top_career_event.NumOfStudentsAttended.values}, index= top_career_event.Topic.values)

    print("Top 5 successful events based on student attendance")
    df.plot(kind='pie', subplots=True, figsize=(16,10))
    plt.show()

#Close the connection:
closeconnection(connection,cursor)
```

Output to Python Code:

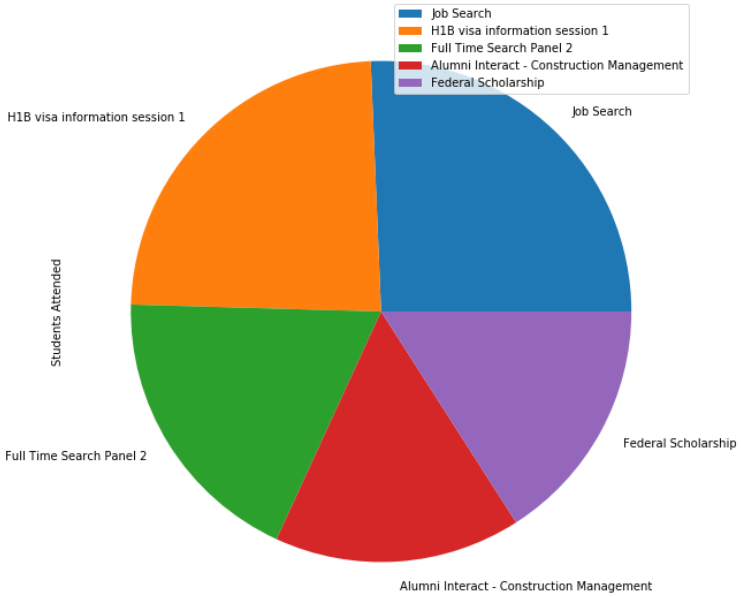
Frequency of events that occurred at each location over the years.



Proportion of job postings of each company



Top 5 successful events based on student attendance



VII. Summary and recommendation

Because of this effective relational model, we were easily able to segregate the job postings, referral system, discussion forums by department, company, alumni etc. This model helps the alumni to easily post about the job offering where it reaches the desired set of students in the institution as the data is well designed throughout the tables with necessary attributes. And students can easily contact the alumni for suggestions and referrals in their desired field of interest.

Furthermore, if we include the attributes like student's area of interests, Demographic preferences, skill set segregation it can benefit the students in search for the placement of current students.