

# IT Alumni Database Milestone 2 Report IT 4983

Ricky Parks

Vy Duong

Zack Downing  
Desiree Smokes

# Summary of Milestone 2 tasks

Task	Status
Connect to database using Python	Completed
Collect and parse data	Completed
Create logic to check if record exists	In progress
Work on logic to insert data	In progress
Testing the insertion logic	Completed

The image shows a PyCharm IDE window with a Python script named 'Insert.py'. The script is located at 'C:\Users\hakee\Desktop\Insert'. The script imports 'json', 'pyodbc', and 'glob' from the 'openpyxl' module. It defines a server variable 'server' with the value 'itcapstone.database.windows.net'. It then connects to a database using 'pyodbc.connect()' with the following parameters: 'Driver=(ODBC Driver 13 for SQL Server);Server=top:itcapstone.database.windows.net,1433;Database=CAPSTONE;Uid=capstone@itcapstone;Pwd=Alumnidata'. A cursor is created using 'cursor = cmxn.cursor()'. Finally, a class 'Alumni' is defined with attributes: 'first\_name', 'last\_name', 'school\_name', 'id', 'degree', 'graduation\_date', 'linked\_in', and 'location'. The IDE interface includes a project explorer on the left, a search bar, and a run console at the bottom. The Windows taskbar is visible at the bottom of the screen.

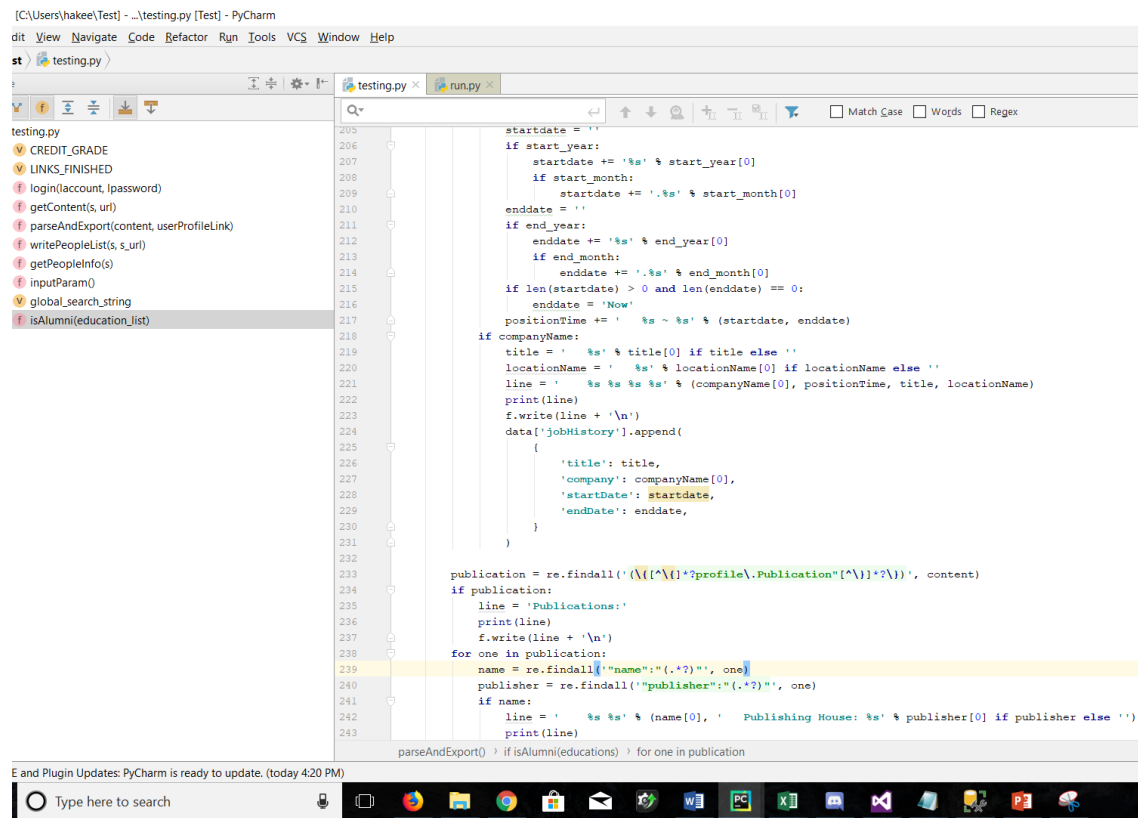
```
1 import json
2 import pyodbc
3 import glob
4
5 from openpyxl import load_workbook
6
7
8 server = 'itcapstone.database.windows.net'
9
10 cmxn = pyodbc.connect('Driver=(ODBC Driver 13 for SQL Server);Server=top:itcapstone.database.windows.net,1433;Database=CAPSTONE;Uid=capstone@itcapstone;Pwd=Alumnidata')
11 cursor = cmxn.cursor()
12
13
14
15
16 class Alumni:
17     first_name = ""
18     last_name = ""
19     school_name = ""
20     id = ""
21     degree = ""
22     graduation_date = ""
23     linked_in = ""
24     location = ""
```

# Connecting to database from Python Script

# Collecting and parsing data

- ▶ Original Web crawler outputs data into .txt files
- ▶ Had to be modified to use JSON
- ▶ We were originally going to use Excel, but we found JSON made it easier to read the data
- ▶ Since Python naturally understands Dictionary objects, this was much simpler and faster

# Collecting and parsing data (continued)



```
[C:\Users\hakeel\Test] - \testing.py [Test] - PyCharm
File View Navigate Code Refactor Run Tools VCS Window Help
testing.py
CREDIT_GRADE
LINKS_FINISHED
login(account, password)
getContent(s, url)
parseAndExport(content, userProfileLink)
writePeopleList(s, s_url)
getPeopleInfo(s)
inputParam()
global_search_string
isAlumni(education_list)

startdate = ''
if start_year:
    startdate += '%s' % start_year[0]
    if start_month:
        startdate += '%s' % start_month[0]
enddate = ''
if end_year:
    enddate += '%s' % end_year[0]
    if end_month:
        enddate += '%s' % end_month[0]
if len(startdate) > 0 and len(enddate) == 0:
    enddate = 'Now'
positionTime += '%s ~ %s' % (startdate, enddate)
if companyName:
    title = '%s' % title[0] if title else ''
    locationName = '%s' % locationName[0] if locationName else ''
    line = '%s %s %s' % (companyName[0], positionTime, title, locationName)
    print(line)
    f.write(line + '\n')
    data['jobHistory'].append(
        {
            'title': title,
            'company': companyName[0],
            'startDate': startdate,
            'endDate': enddate,
        }
    )

publication = re.findall('([^\s]*?profile\.Publication"[^"]*"?)', content)
if publication:
    line = 'Publications:'
    print(line)
    f.write(line + '\n')
    for one in publication:
        name = re.findall('"name": "(.*)"', one)
        publisher = re.findall('"publisher": "(.*)"', one)
        if name:
            line = '%s %s' % (name[0], 'Publishing House: %s' % publisher[0] if publisher else '')
            print(line)

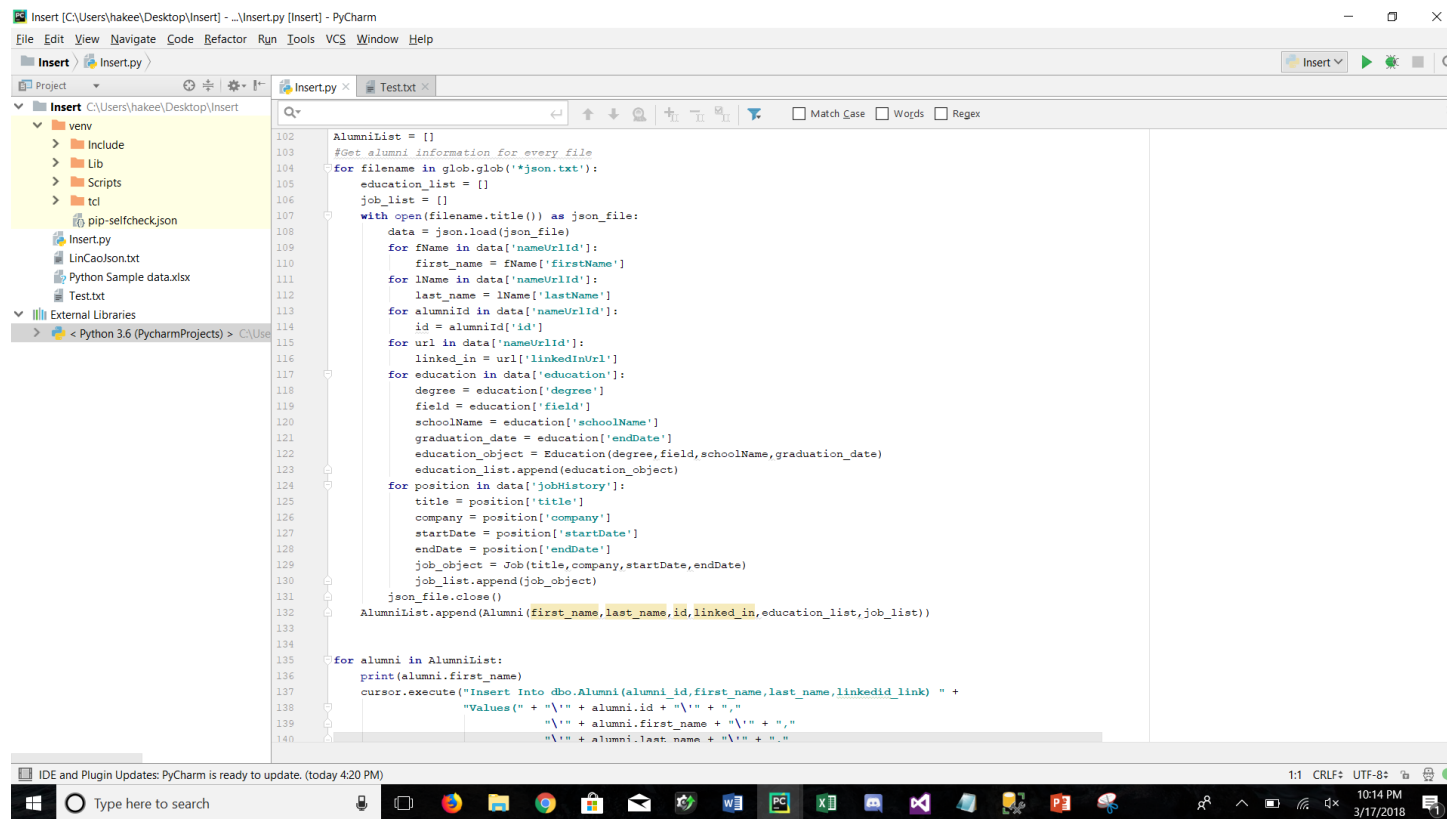
parseAndExport()
if isAlumni(educations):
    for one in publication
```

- Highlighted section shows the web crawler modified to use JSON

# Checking if record exists before insert

- ▶ Currently the primary key attributes in each table prevents duplicate data
- ▶ It's still being determined as to whether or not we need this functionality

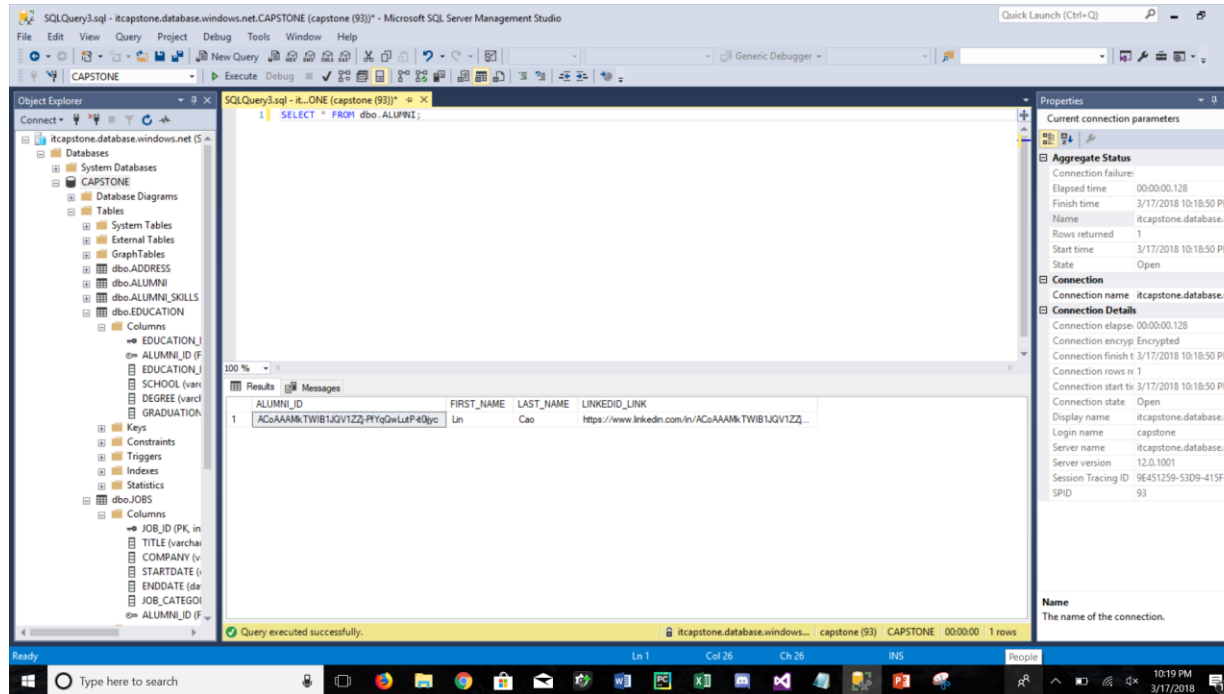
# Inserting data into database from web crawler files



The screenshot displays the PyCharm IDE interface. The left sidebar shows a project structure with folders like 'venv', 'Include', 'Lib', 'Scripts', and 'tcl', along with files like 'Insert.py', 'LinCaolson.txt', 'Python Sample data.xlsx', and 'Test.txt'. The main editor window shows a Python script named 'Insert.py' with the following code:

```
102 Alumnist = []
103 #Get alumni information for every file
104 for filename in glob.glob('*json.txt'):
105     education_list = []
106     job_list = []
107     with open(filename, 'r') as json_file:
108         data = json.load(json_file)
109         for fName in data['nameUrlId']:
110             first_name = fName['firstName']
111             lName = data['nameUrlId']
112             last_name = lName['lastName']
113             for alumniId in data['nameUrlId']:
114                 id = alumniId['id']
115                 for url in data['nameUrlId']:
116                     linked_in = url['linkedInUrl']
117                 for education in data['education']:
118                     degree = education['degree']
119                     field = education['field']
120                     schoolName = education['schoolName']
121                     graduation_date = education['endDate']
122                     education_object = Education(degree, field, schoolName, graduation_date)
123                     education_list.append(education_object)
124                 for position in data['jobHistory']:
125                     title = position['title']
126                     company = position['company']
127                     startDate = position['startDate']
128                     endDate = position['endDate']
129                     job_object = Job(title, company, startDate, endDate)
130                     job_list.append(job_object)
131             json_file.close()
132             Alumnist.append(Alumni(first_name, last_name, id, linked_in, education_list, job_list))
133
134 for alumni in Alumnist:
135     print(alumni.first_name)
136     cursor.execute("Insert Into dbo.Alumni (alumni_id, first_name, last_name, linkedid_link) " +
137                    "Values (" + "\"" + alumni.id + "\"" + ", " +
138                    "\"" + alumni.first_name + "\"" + ", " +
139                    "\"" + alumni.last_name + "\"" + ", " +
```

The bottom status bar indicates the IDE and Plugin Updates: PyCharm is ready to update. (today 4:20 PM). The taskbar at the bottom shows the Windows Start button, a search bar, and various application icons. The system clock shows 10:14 PM on 3/17/2018.



Data from Alumni table after being inserted from Python



# Lessons learned

- ▶ We learned a lot about the Python language and libraries
- ▶ Got more familiar with parsing using JSON
- ▶ Learned that we have to come up with an optimal rate for running the web crawler to avoid sending out too many page requests
- ▶ We may have to use a premium business LinkedIn account to be granted a higher number of monthly searches on the LinkedIn

# Plan going Forward

- ▶ Research the optimal number of page requests to send to LinkedIn
- ▶ Determine the best way to update the database using the LinkedIn urls
- ▶ Begin documenting all of the scripts involved
- ▶ Analyze the scripts to see if they can be made to run faster