

Name: RAVITEJA KASE #ID: B00823644

Disclaimer (Data Scraping)

In assignment 1 of CSCI 5408 course, data scraping is done manually or programmatically from Dalhousie University's website, and it is used only for educational purpose. Sensitive information, such as personal Email, personal contact numbers are not extracted. However, names of instructors, professors, or other staff members available on the Dalhousie University websites are extracted for course (CSCI 5408) related analysis, such as "find how many employees have similar first name etc." The scope of the extracted data usage is limited to the course CSCI 5408 only. The course instructor and the Faculty of Computer Science cannot be held responsible for any misuse of the extracted data.

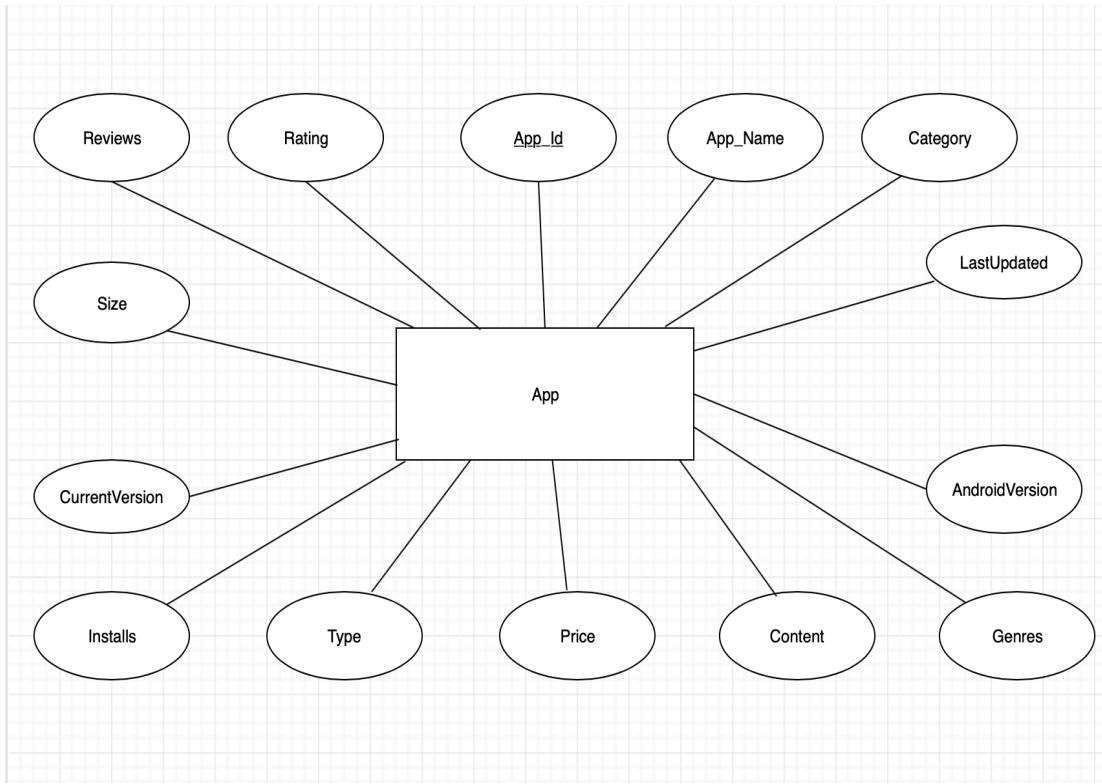
A. Data Modelling (Initial Design or the Conceptual Model)

1. Initial data model from Google App Ratings Data:

Based on the data available in the googleplaystore.csv, entity and attributes are as follows.[2]

Entity: App

Attributes: AppId (PrimaryKey), Category, Rating, Reviews, Size, Installs, Type, Price, Content, Genres, LastUpdated, CurrentVersion, AndroidVersion.



2. Initial Data model from DAL website:

The objective is to design an information system for people who want to join the university as students, faculty member, or staff. Here are the observed entities from the DAL website.[2]

1. Entity: Department
Attributes: DepartmentId (Primary Key), DepartmentName
2. Entity: Employee
Attributes: EmpId (Primary Key), LastName, FirstName, DepartmentId
3. Entity: UnderGraduatePrograms
Attributes: UGID (Primary Key), UgName, DepartmentId
4. Entity: GraduatePrograms
Attributes: GraduateID (Primary Key), GraduateProgramName, DepartmentId
5. Entity: Courses
Attributes: CourseId (Primary Key) ,CourseName , ProgramId
6. Entity: Buildings
Attributes: BuildingId (Primary Key), BuildingName, Location, LoungeArea, StudySpace, FoodServices, Wifi
7. Entity: Research
Attributes: ResearchId (Primary Key), ResearchArea, ResearchInstitute
8. Entity: Residencies

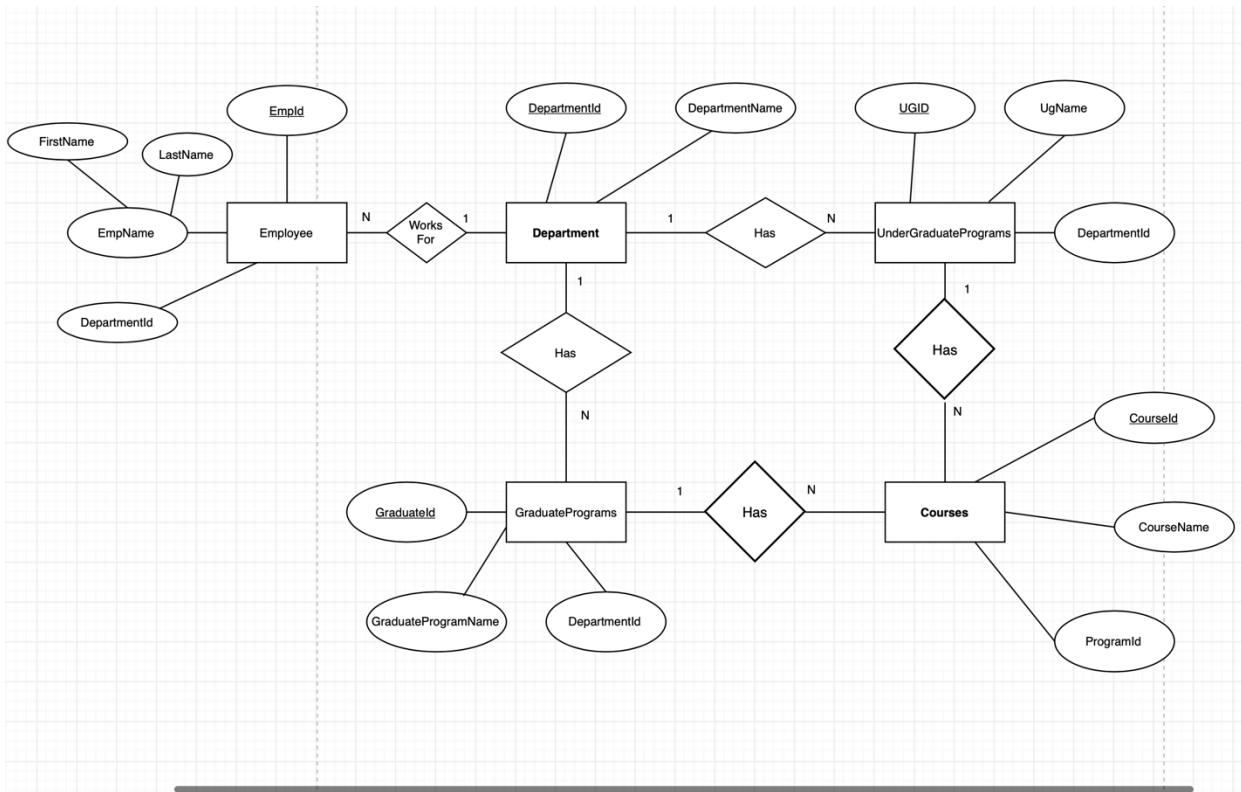
Attributes: ResidencyID (Primary Key), ResidencyBuildingName, RoomTypeId, ResidencyType, Location, FallFees, WinterFees, TotalFees

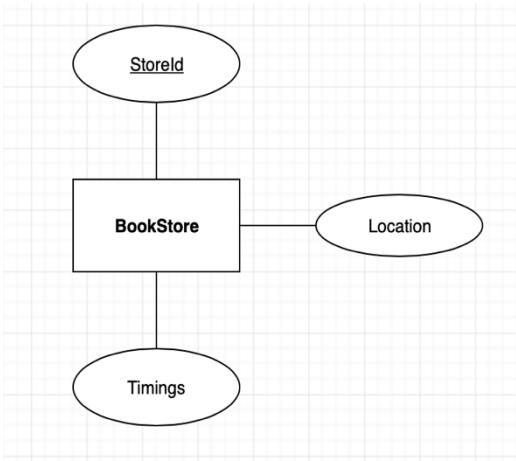
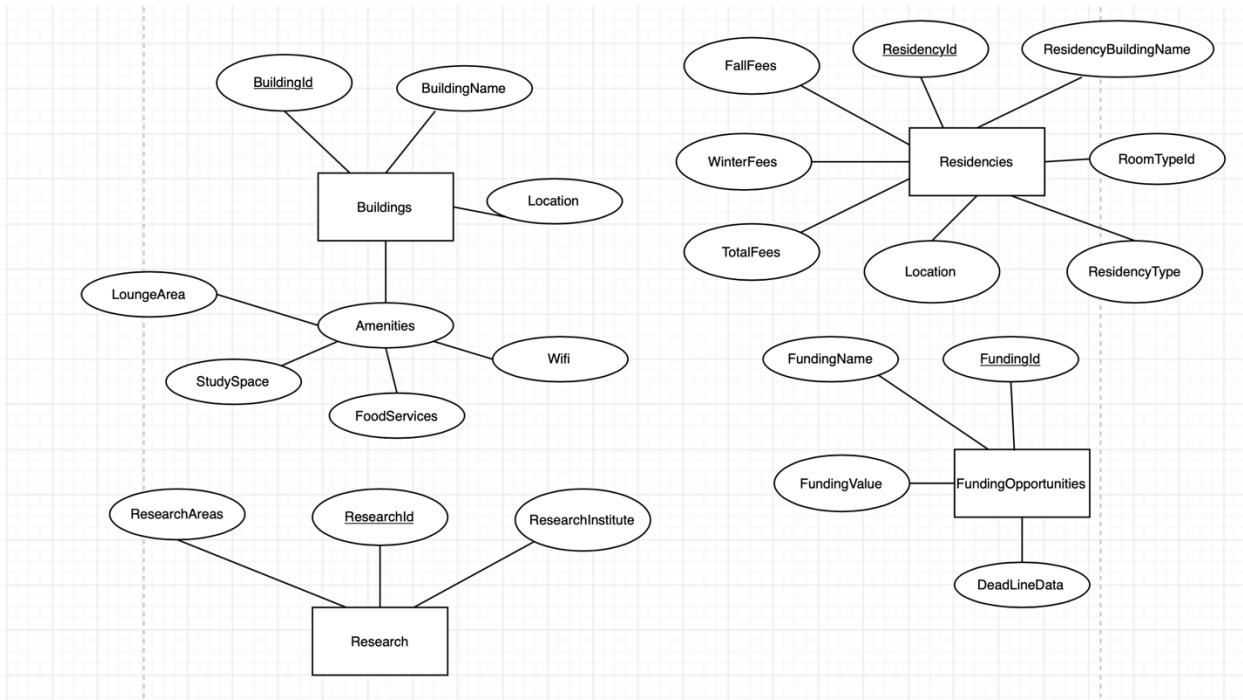
9. Entity: FundingOpportunities

Attributes: FundingId (Primary Key), FundingName, FundingValue, DeadlineData

10. Entity: BookStore

Attributes: StoreId (Primary Key), StoreName, Location





B. Data extraction and Data Collection.

- Duplicate rows from the googleplaystoe.csv are removed and inserted into the KaggleApp database and into App table.

3. Script to extract Data from DAL website and store it in XML Format

Script is in PythonProgram-In_Src_Folder /src/. folder

Script – Dal_information.py

Xml files:

- Departments.xml
- UnderGraduatePrograms.xml
- GraduatePrograms.xml

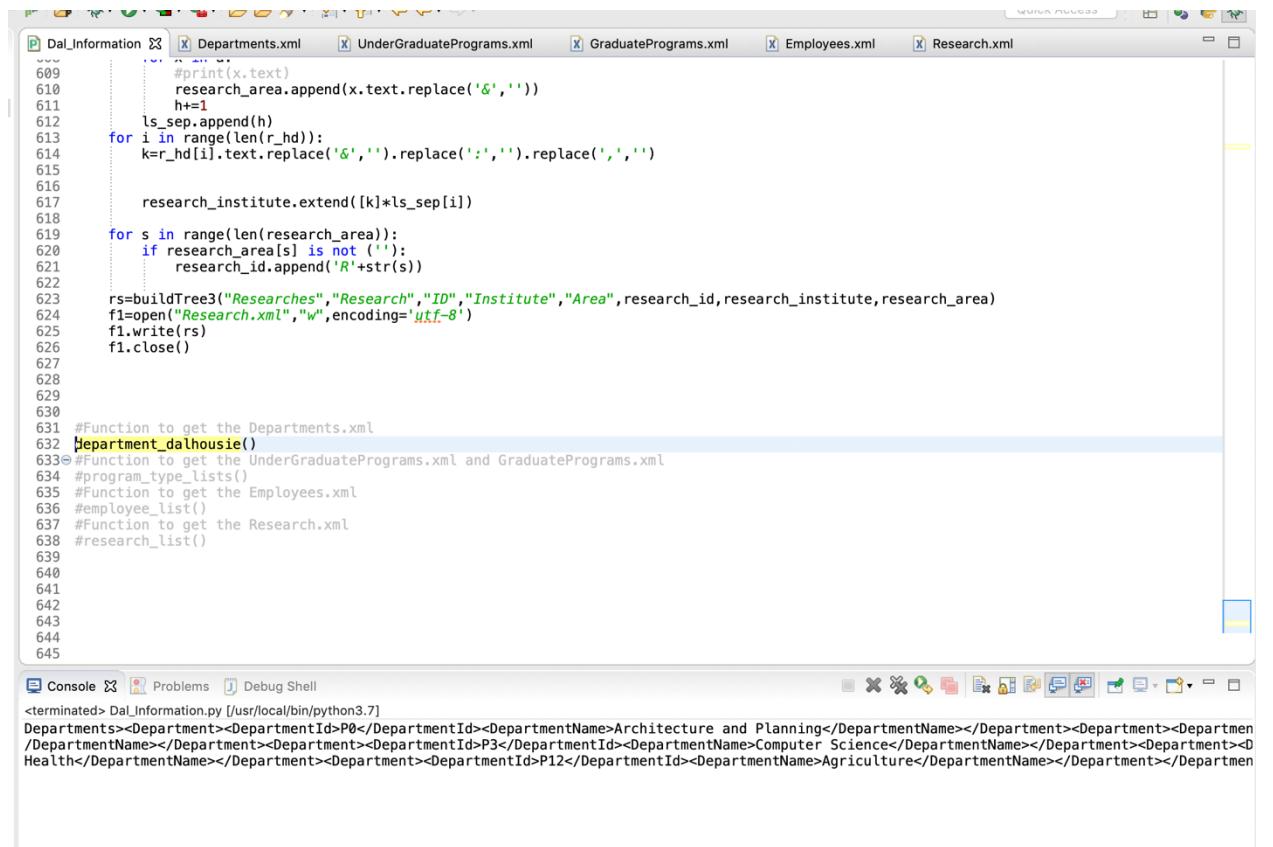
- Employees.xml
- Research.xml

Libraries used:

- import requests
- from bs4 import BeautifulSoup[1]

department_dalhousie() - Collects the data for faculties , faculty names are fetched and Ids are auto generated and loaded into Departments.xml file.

Departments.xml contains DepartmentId, DepartmentName



```

609     #print(x.text)
610     research_area.append(x.text.replace('&',''))
611     h+=1
612     ls_sep.append(h)
613     for i in range(len(r_hd)):
614         k=r_hd[i].text.replace('&','').replace(':','').replace(' ','')
615
616         research_institute.extend([k]*ls_sep[i])
617
618     for s in range(len(research_area)):
619         if research_area[s] is not (''):
620             research_id.append('R'+str(s))
621
622     rs=buildTree3("Researches","Research","ID","Institute","Area",research_id,research_institute,research_area)
623     f1=open("Research.xml","w",encoding='utf-8')
624     f1.write(rs)
625     f1.close()
626
627
628
629
630
631 #Function to get the Departments.xml
632 def department_dalhousie():
633     #Function to get the UnderGraduatePrograms.xml and GraduatePrograms.xml
634     #program_type_lists()
635     #Function to get the Employees.xml
636     #employee_list()
637     #Function to get the Research.xml
638     #research_list()
639
640
641
642
643
644
645

```

Console output:

```

<terminated> Dal_Information.py [/usr/local/bin/python3.7]
Departments><Department><DepartmentId>P0</DepartmentId><DepartmentName>Architecture and Planning</DepartmentName></Department><Department><DepartmentName></DepartmentName></Department><Department><DepartmentId>P3</DepartmentId><DepartmentName>Computer Science</DepartmentName></Department><Department><DepartmentName>Health</DepartmentName></Department><DepartmentId>P12</DepartmentId><DepartmentName>Agriculture</DepartmentName></Department></Departmen

```

program_type_lists()- Collects the undergraduate and graduate programs from specific faculties and auto generates the UGID, graduated and loads into UnderGraduatePrograms.xml and GraduatePrograms.xml.

For the following faculties, UG and graduate programs are collected:

Architecture and planning, Arts and Science, Computer Science, Dentistry, Engineering, Health, Management, Medicine, Science, Agriculture.

UnderGraduatePrograms.xml contains UGID, UnderGraduateProgramName, DepartmentID

```

610     #print(x.text)
611     research_area.append(x.text.replace('&', ''))
612     h+=1
613     ls_sep.append(h)
614     for i in range(len(r_hd)):
615         k=r_hd[i].text.replace('&','').replace(':','').replace(',','')
616
617         research_institute.extend([k]*ls_sep[i])
618
619     for s in range(len(research_area)):
620         if research_area[s] is not (''):
621             research_id.append('R'+str(s))
622
623     rs=buildTree3("Researches","Research","ID","Institute","Area",research_id,research_institute,research_area)
624     f1=open("Research.xml","w",encoding='utf-8')
625     f1.write(rs)
626     f1.close()
627
628
629
630
631
632 #Function to get the Departments.xml
633 #department_dalhousie()
634 #Function to get the UnderGraduatePrograms.xml and GraduatePrograms.xml
635 program_type_lists()
636 #Function to get the Employees.xml
637 #employee_list()
638 #Function to get the Research.xml
639 #research_list()
640
641
642
643
644
645
646

```

<terminated> Dal_Information.py [/usr/local/bin/python3.7]

<UnderGraduatePrograms><UG><UGID>B0</UGID><UnderGraduateProgramName> Bachelor of Environmental Design Studies – BEDS</UnderGraduateProgramName><Depa</UnderGraduateProgramName><DepartmentId>P11</DepartmentId><UG><UGID>B63</UGID><UnderGraduateProgramName>Pharmacy (BSc)</UnderGraduteProgramNa</UnderGraduateProgramName><DepartmentId>P11</DepartmentId><UG><UGID>B68</UGID><UnderGraduateProgramName>Therapeutic Recreation (BSc)</UnderGr

GraduatePrograms.xml contains GraduateID, GraduateProgramName, DepartmentID

```

610     #print(x.text)
611     research_area.append(x.text.replace('&', ''))
612     h+=1
613     ls_sep.append(h)
614     for i in range(len(r_hd)):
615         k=r_hd[i].text.replace('&','').replace(':','').replace(',','')
616
617         research_institute.extend([k]*ls_sep[i])
618
619     for s in range(len(research_area)):
620         if research_area[s] is not (''):
621             research_id.append('R'+str(s))
622
623     rs=buildTree3("Researches","Research","ID","Institute","Area",research_id,research_institute,research_area)
624     f1=open("Research.xml","w",encoding='utf-8')
625     f1.write(rs)
626     f1.close()
627
628
629
630
631
632 #Function to get the Departments.xml
633 #department_dalhousie()
634 #Function to get the UnderGraduatePrograms.xml and GraduatePrograms.xml
635 program_type_lists()
636 #Function to get the Employees.xml
637 #employee_list()
638 #Function to get the Research.xml
639 #research_list()
640
641
642
643
644
645
646

```

<terminated> Dal_Information.py [/usr/local/bin/python3.7]

<GraduatePrograms><GraduateProgram><GraduateId>M0</GraduateId><GraduateProgramName>Master of Architecture – MArch</GraduateProgramName><DepartmentId><GraduateProgramName><DepartmentId>P11</DepartmentId><GraduateProgram><GraduateId>M45</GraduateId><GraduateProgramName>Pharmaceut</GraduateProgramName><DepartmentId>P11</DepartmentId><GraduateProgram><GraduateProgram><GraduateId>M49</GraduateId><GraduateProgramName>Social Wor</GraduateProgramName><DepartmentId>P11</DepartmentId><GraduateProgram><GraduateProgram><GraduateId>M50</GraduateId><GraduateProgramName>Speech Lar<(graduate and postdoctoral opportunities)</GraduateProgramName><DepartmentId>P9</DepartmentId><GraduateProgram><GraduateProgram><GraduateId>M63</C

employee_list() -Collects the names of staff in each faculty and auto generates Id for each employee and loads into Employees.xml.

For the following faculties, employee details are collected:

- Staff in Agriculture (Animal science agriculture)
- staff in Architecture and planning
- staff in Arts and science
- faculty-staff in computer science
- faculty-staff in Engineering (Electrical)
- staff in health

Employees.xml contains EmpId, FirstName, LastName, Department

The screenshot shows a code editor with a Python script named `Dal_Information.py`. The script contains several functions: `research_id`, `research_institute`, `buildTree3`, `department_dalhousie`, `UnderGraduatePrograms.xml`, `GraduatePrograms.xml`, `employee_list`, and `research_list`. The `employee_list` function is highlighted. The console tab at the bottom shows the execution of the script and the resulting XML output. The XML output lists employees with their first names, last names, department IDs, and research areas.

```
611     #print(x.text)
612     research_area.append(x.text.replace('&', ''))
613     h+=1
614     ls_sep.append(h)
615     for i in range(len(r_hd)):
616         k=r_hd[i].text.replace('&', '').replace('::','').replace(',','')
617
618         research_institute.extend([k]*ls_sep[i])
619
620     for s in range(len(research_area)):
621         if research_area[s] is not (''):
622             research_id.append('R'+str(s))
623
624     rs=buildTree3("Researches","Research","ID","Institute","Area",research_id,research_institute,research_area)
625     f1=open("Research.xml","w",encoding='utf-8')
626     f1.write(rs)
627     f1.close()
628
629
630
631
632
633#Function to get the Departments.xml
634 #department_dalhousie()
635 #Function to get the UnderGraduatePrograms.xml and GraduatePrograms.xml
636 #program_type_lists()
637 #Function to get the Employees.xml
638 employee_list()
639#Function to get the Research.xml
640 #research_list()
641
642
643
644
645
646
647
```

```
<terminated> Dal_Information.py [/usr/local/bin/python3.7]
<Employees><Employee><EmpId>E0</EmpId><FirstName>Donna</FirstName><LastName>Jamieson</LastName><Department>P12</Department></Employee><Employee><EmpId>E1</EmpId><FirstName>Marilyn</FirstName><LastName>Roberts</LastName><Department>P0</Department><Employee><EmpId>E4</EmpId><FirstName>Ken</FirstName><LastName>Kam</LastName><Department>P0</Department></Employee><Employee><EmpId>E6</EmpId><FirstName>Ken</FirstName><LastName>Kam</LastName><Department>P0</Department><Employee><EmpId>E7</EmpId><FirstName>Christina</FirstName><LastName>MacNeil</LastName><Department>P0</Department><Employee><EmpId>E13</EmpId><FirstName>Jeremy</FirstName><LastName>Sutherland</LastName><Department>P1</Department><Employee><EmpId>E17</EmpId><FirstName>Ken</FirstName><LastName>Kam</LastName><Department>P1</Department><Employee><EmpId>E18</EmpId><FirstName>Christina</FirstName><LastName>MacNeil</LastName><Department>P1</Department><Employee><EmpId>E24</EmpId><FirstName>Jeremy</FirstName><LastName>Sutherland</LastName><Department>P3</Department><Employee><EmpId>E66</EmpId><FirstName>Yannick</FirstName><LastName>Marchand</LastName><Department>P5</Department><Employee><EmpId>E138</EmpId><FirstName>Peter</FirstName><LastName>Greson
```

research_list() -Collects the research institutes and research areas done in each center and loads into Research.xml

Research.xml contains ResearchId, ResearchInstitute, ResearchArea

The screenshot shows a Python development environment with several tabs open at the top: 'Dal_Information', 'Departments.xml', 'UnderGraduatePrograms.xml', 'GraduatePrograms.xml', 'Employees.xml', and 'Research.xml'. The main code editor window displays a script named 'Dal_Information.py' containing Python code for processing XML files. The code includes functions for reading 'Departments.xml', 'UnderGraduatePrograms.xml', 'GraduatePrograms.xml', 'Employees.xml', and 'Research.xml', and building a tree structure for research areas. A specific section of the code is highlighted in blue. Below the code editor is a 'Console' tab showing the output of the script's execution. The console output shows the script was terminated successfully and lists the XML elements it processed: '<Researches><Research><ID>R0</ID><Institute>Sustainable Ocean</Institute><Area>Canadian Institute of Fisheries Technology </Area></Research><Research'.

C. Data Insertion:

Dal_1 database is created in MySQL and the following tables were created and the respective xml files are used for insertion.

Department table -(Departments.xml)

Employee table-(Employees.xml)

UnderGraduatePrograms table -(UnderGraduatePrograms.xml)

GraduatePrograms table-(GraduatePrograms.xml)

Research table -(Research.xml)

Course table

Residencies table

FundingOpportunities table

BookStore table

Buildings table

D. Final Data Modeling:

1. Is your initial sketch/design free from any design issues?

No, Fan trap design issue is present.[3]

Department entity is in three 1:M relationships (Employee, UnderGraduatePrograms, GraduatePrograms).

Justification: According to the design, an employee belongs to a specific department. Each department has particular undergraduate programs are graduate programs.

Each employee who works for particular department will work for all the undergraduate programs as well as graduate programs falling under the department. For example, professor can teach for MACS as well as MCS. Similar way administrative staff also works for all the programs in a particular department.

So, it is not required to have one- one mapping for Employee and Undergraduate Programs and Employee and Graduate Programs.

If queried, we can be able to fetch the employee names who work for a particular undergraduate program.

The below query fetches the employee names who work for “International Food Business Program”.

Similarly, if queried we can be able to fetch the undergraduate programs which belong to particular department, where employee works.

NOTE: In the website, faculty and staff list also provided for specific departments. So the design with Fan traps would not cause any problem.

Final Design:

1. Identify the relationships, and cardinality between the entity sets you created.
 - DepartmentId from Department is referred in UndergradatPrgrms. Department has UnderGraduateProgramas [1:M] relationship
 - DepartmentId from Department is referred in GraduatePrograms. Department has GraduatePrograms [1:M] relationship
 - DepartmentId from Department is referred in Employees. Department has Employees [1:M] relationship
 - UGID from UnderGraduatePrograms is referred in Courses as ProgramId UnderGraduatePrograms has courses [1:M] relationship
 - GraduateID from GraduatePrograms is referred in Courses as ProgramId GraduatePrograms has courses [1:M] relationship

2. Construct an extended ERD. Your ERD should highlight if any overlap/disjoint subtype exists.

Following ERD is generated and is present in the images folder. ERD doesn't have overlap/disjoint subtypes.

3. The extended ERD should be created using a tool, such as MySQL workbench/ ErWin/ Visio etc.,

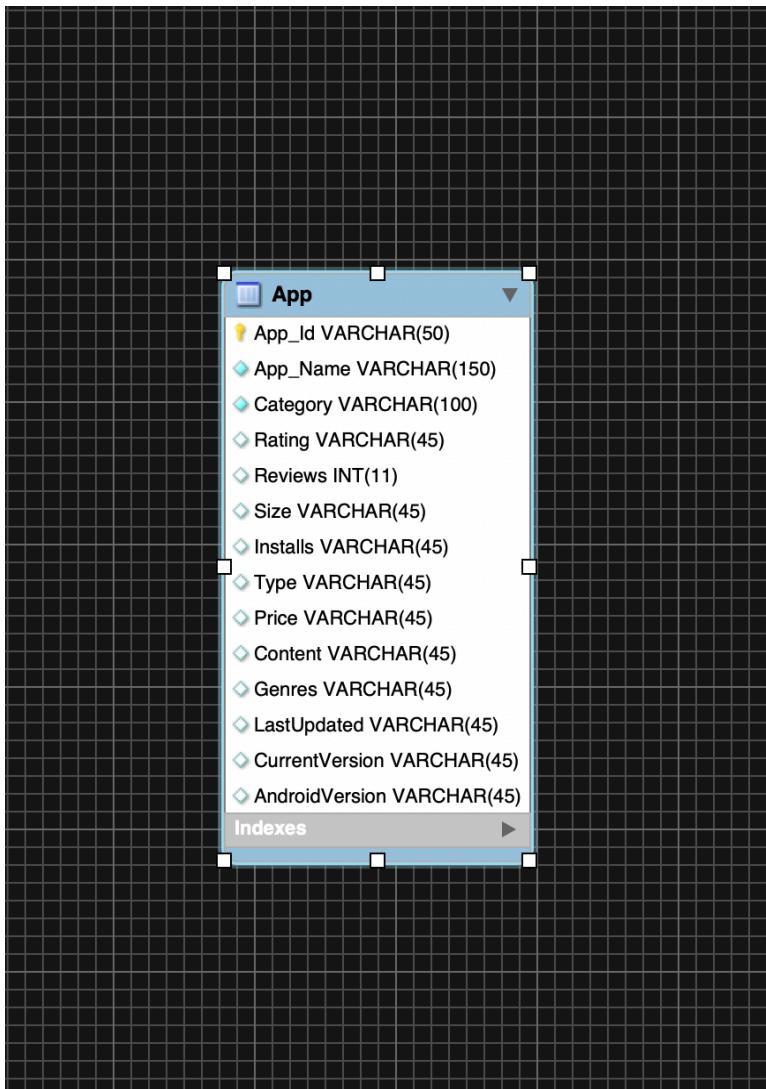
Extended ERD is generated using MySQL workbench.

E. Normalization:

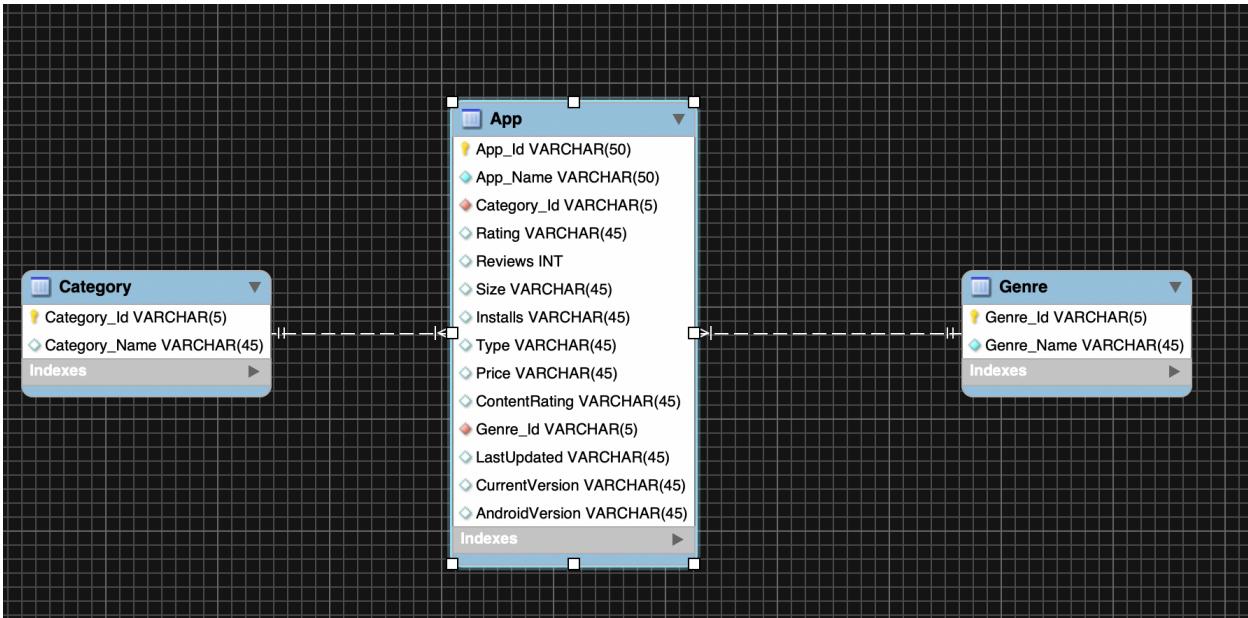
Google Play Store data:

App table is in 2NF with no transitive dependencies. However, the category and genres of app can increase at some point of time. So, App table is broken down into App table, Category table (CategoryId, Category name) and Genre (Genreid, Genre name)

Before Normalization:



After Normalization:



Dalhousie Website:

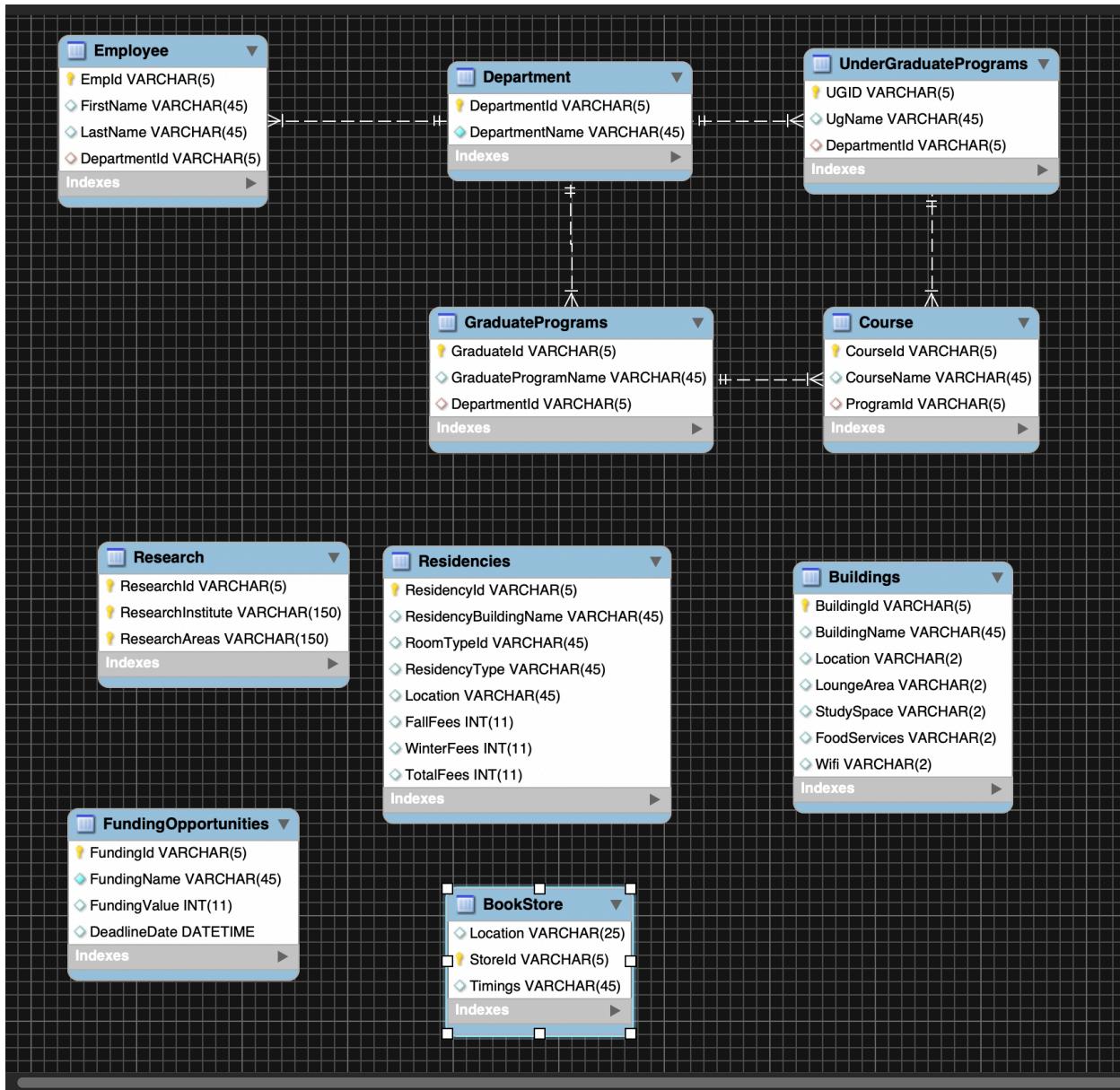
As while data extraction from the website, data was collected accordingly. For example, departments data is collected from and stored. Similarly, data is collected for undergraduate programs, graduate programs, employees table.

Normalization is done for Research table

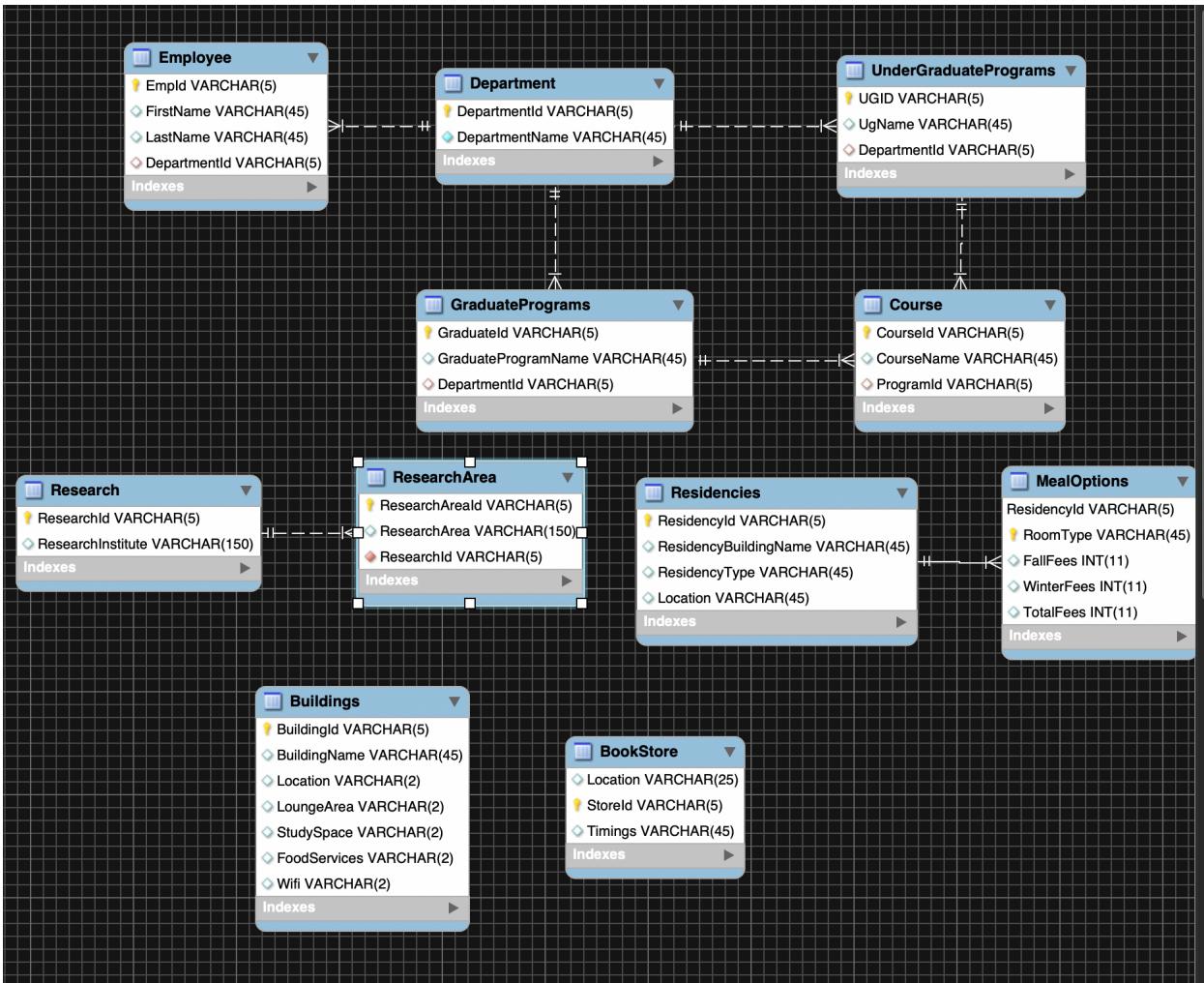
Research table have ResearchId, Research Institute and Research Area. So, research Area is partially dependent on both research Institute and ResearchId.

So, tables were broken down to create a 3NF form. Research table contains Research Id and ResearchInstitute and Research Area table contains Area Id, Research Area and Research ID which is a foreign key and refers Research table.

Before Normalization:



After Normalization:



F. Write SQL Query:

As the queries required Department, Employees , Undergraduate programs , these tables were not required to further normalization , same queries would fetch the results.

- Find the name of the department or faculty that has the highest number of employees having last name starting with an "A"

```
#GET THE DEPARTMENT NAME WHICH HAS MORE NUMBER OF EMPLOYEES WHOSE LAST NAME START WITH 'A'
```

```
SELECT DEPARTMENTID,DEPARTMENTNAME FROM DEPARTMENT WHERE DEPARTMENTID IN (
    SELECT DEPARTMENTID FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE 'A%' GROUP BY DEPARTMENTID HAVING COUNT(DEPARTMENTID)=(SELECT MAX(E_MAX) FROM (SELECT DEPARTMENTID,COUNT(DEPARTMENTID) AS E_MAX FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE 'A%' GROUP BY DEPARTMENTID) A))
```

Answer is Computer Science .

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel (Schemas):** Shows the schema structure for 'App_1'. The 'Employee' table is selected.
- Query Editor:** Displays the SQL query:

```
1  #GET THE DEPARTMENT NAME WHICH HAS MORE NUMBER OF EMPLOYEES WHOSE LAST NAME START WITH 'A'
2  SELECT DEPARTMENTID,DEPARTMENTNAME FROM DEPARTMENT WHERE DEPARTMENTID IN (
3  SELECT DEPARTMENTID FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE 'A%' GROUP BY DEPARTMENTID HAVING COUNT(DEPARTMENTID)=(SELECT MAX(E_MAX) FROM
4  (SELECT DEPARTMENTID,COUNT(DEPARTMENTID) AS E_MAX FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE 'A%' GROUP BY DEPARTMENTID) A)
```
- Result Grid:** Shows the result of the query, which is a single row:

DEPARTMENTID	DEPARTMENTNAME
P3	Computer Science
- Action Output:** Shows the execution log with 60 entries, detailing each step of the query execution.
- Bottom Status:** 'Query Completed'

2. Find the name of the department or faculty that has the highest number of undergraduate programs

#GET THE DEPARTMENT WHICH HAS MORE UNDERGRADUATE PROGRAMS

```
SELECT DEPARTMENTID,DEPARTMENTNAME FROM DEPARTMENT WHERE
DEPARTMENTID=
(SELECT DEPARTMENTID FROM UNDERGRADUATEPROGRAMS GROUP BY
DEPARTMENTID HAVING COUNT(DEPARTMENTID)=(SELECT MAX(MAX_D) FROM
(SELECT DEPARTMENTID,COUNT(DEPARTMENTID)AS MAX_D FROM
UNDERGRADUATEPROGRAMS GROUP BY DEPARTMENTID)A));
```

Answer is Arts and Social Science

```

1  #GET THE DEPARTMENT WHICH HAS MORE NUMBER OF UNDERGRADUATE PROGRAMS
2  SELECT DEPARTMENTID,DEPARTMENTNAME FROM DEPARTMENT WHERE DEPARTMENTID =1
3  SELECT DEPARTMENTID FROM UNDERGRADUATEPROGRAMS GROUP BY DEPARTMENTID HAVING COUNT(DEPARTMENTID)=(SELECT MAX(MAX_D) FROM
4  (SELECT DEPARTMENTID,COUNT(DEPARTMENTID) AS MAX_D FROM UNDERGRADUATEPROGRAMS GROUP BY DEPARTMENTID) A)
5
6
7

```

Object Info	Session	DEPARTMENTID	DEPARTMENTNAME
P1		10	Arts and Social Sciences

Action Output	Time	Action	Response	Duration / Fetch Time
51	17:53:44	PREPARE stmt FROM 'INSERT INTO `App_1`.`employee` (`empId` , `LastName`) VALUES (? , ?)'	OK	0.000 sec
52	17:53:45	DEALLOCATE PREPARE stmt;	OK	0.000 sec
53	17:54:00	select * from employee LIMIT 0, 1000	163 row(s) returned	0.0010 sec / 0.00002...
54	17:58:05	select * from employee where lower(lastname) like 'a%' LIMIT 0, 1000	10 row(s) returned	0.0051 sec / 0.00001...
55	17:58:58	SELECT DEPARTMENTID FROM EMPLOYEES WHERE UPPER(LASTNAME) LIKE '%A%'	Error Code: 1146. Table 'app_1.employees' doesn't exist	0.0024 sec
56	17:59:08	SELECT DEPARTMENTID FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE '%A%'	10 row(s) returned	0.00039 sec / 0.000...
57	17:59:44	SELECT DEPARTMENTID FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE '%A%'	2 row(s) returned	0.0038 sec / 0.000...
58	18:00:12	SELECT DEPARTMENTID,COUNT(DEPARTMENTID) AS E_MAX FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE '%A%'	2 row(s) returned	0.00041 sec / 0.000...
59	18:03:29	SELECT DEPARTMENTID FROM EMPLOYEE WHERE UPPER(LASTNAME) LIKE '%A%'	1 row(s) returned	0.00099 sec / 0.000...
60	18:04:07	SELECT DEPARTMENTID,DEPARTMENTNAME FROM DEPARTMENT WHERE DEPARTMENTID =1	1 row(s) returned	0.0037 sec / 0.0000...

References

- [1]. Tutorial: Python Web Scraping Using BeautifulSoup.[Online]. Available: <https://www.dataquest.io/blog/web-scraping-tutorial-python/> . [Accessed 20th September 2019]
- [2]. Data Model:]Ramez Elmasri, Shamkant B. Navathe, “*Fundamentals of Database Systems*”, 7the
- [3]: Fan trap design issue[Online]: DBMS Concepts_ERD_Normalization lecture slide in Brightspace.

