Data Management – Final Project

# Submitted by –Anuj K Yadav

**Title:** **Online Practical Training**

**Problem statement:**

Graduate student relies on Internship opportunities to practice the concepts learned in the course. With the huge volume of students, many students are deprived of the opportunity to practice on desired profile due to lack of a centralized platform to search and apply for employment opportunities. Thus, they lack in work experience in their field and are left with a not so strong portfolio for a desired full-time job. At the same time, employers are not able to derive information on how good a student is for some specific job role due to lack of rating based historical data.

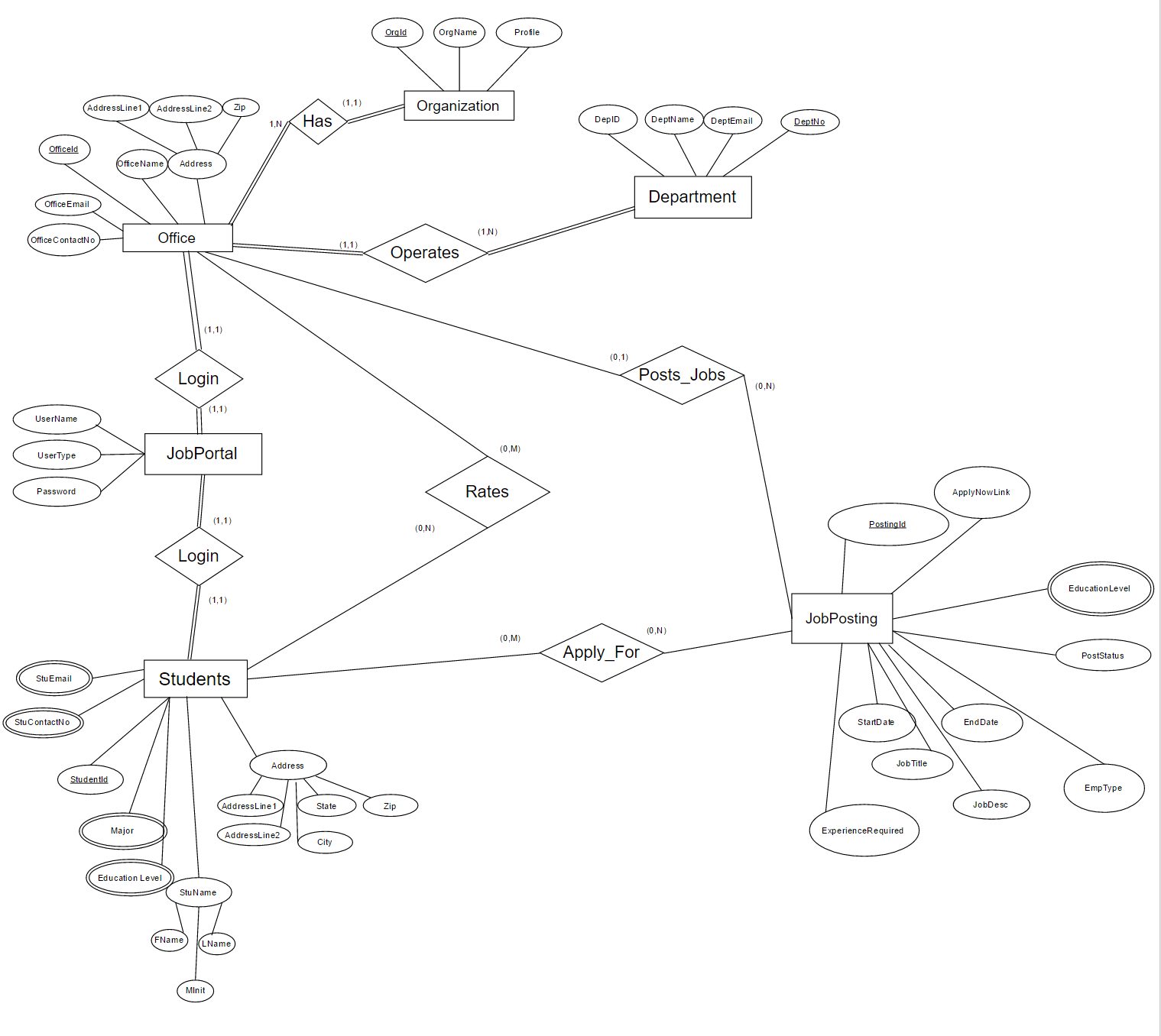
**Proposed Solution/ Project to be Delivered:**

We are trying to build a platform for students to have the access to the relevant jobs posted online by the employers for training purpose. This solves two key issues

1.       It builds confidence of the students on the concepts/tools learned in the course.

2.       The Feedback feature adds credibility to the student by providing them ratings which can be utilized by employers for a better screening process.

**ER Diagram:**



**Data Items, Entities, Attributes and Restrictions:**

* Organization

OrgID (String ; 5)

Org\_Name (String ; 40)

* Office

OfficeID (String ; 5)

Office\_Name (String ; 40)

AddLine1 (String ; 40)

AddLine2 (String ; 40)

ZipCode (Number ; 5)

City (String ; 20)

State (String ; 10)

Country (String ; 12)

* Department

DeptID (String ; 5)

Dept\_Name (String ; 40)

Dept\_Email (String ; 25)

Dept\_ContactNo(Number ; 10)

* Student

StudentID (String ; 5)

Stu\_FName (String ; 20)

Stu\_MName (String ; 10)

Stu\_LName (String ; 20)

AddLine1 (String ; 40)

AddLine2 (String ; 40)

ZipCode (Number ; 5)

City (String ; 20)

State (String ; 10)

Country (String ; 12)

Stu\_Email (String ; 25)

Stu\_ContactNo (Number ; 10)

* Profile

ProfileID (String ; 5)

AboutMe (String ; 400)

AreaOfInterest (String ; 50)

* Job Posting

PostingID (String ; 5)

Job\_Desc (String ; 400)

Emp\_type (String ; 20)

Start\_date (Date ; 10)

End\_Date (Date ; 10)

Status (String ; 10)

Experience (String ; 20)

AppyNowLink (String ; 100)

* Authorization

AuthID (String ; 5)

Username (String ; 20)

Password (String ; 20)

UserType (String ; 3)

* University

University\_Code (String ; 5)

University\_Name (String ; 50)

AddLine1 (String ; 40)

AddLine2 (String ; 40)

ZipCode (Number ; 5)

City (String ; 20)

State (String ; 10)

Country (String ; 12)

* Education

MajorName (String ; 40)

Edu\_level (String ; 20)

* Rating

Rating (Decimal ; 1,1)

**(Min, Max) constraints:**

Min, Max constraints are mentioned in ER diagram and are individually mentioned below:

* Organization Has Offices – (1, N)  
  Office Is in Organization – (1, 1)
* Office Operates Department – (1, N)

Department is Operated by Office – (1, 1)

* Office Posts Job Posting – (0, N)  
  Job Posting Posted by Office – (0, 1)
* Student Applies for Job Posting – (0, N)  
  Job Posting is Applied by Student – (0, N)
* Office Logins to Job Portal – (1,1)  
  Student Logins to Job Portal – (1,1)
* Office Rates Students – (0, N)  
  Student is Rated by Office – (0, N)

**Cardinality constraints:**

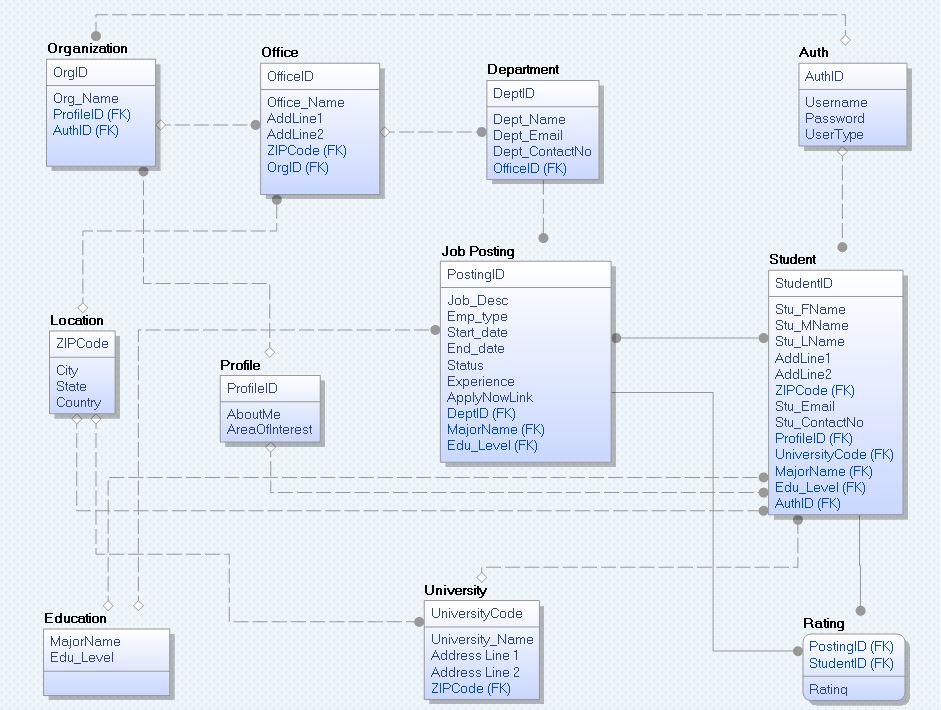
Below are the various cardinality constraints:

* Organization – Office - 1:N
* Office – Department - 1:N
* Office – Job Portal - 1:1
* Student – Job Portal - 1:1
* Office – Job Posting – 1:N
* Student – Job Posting – M:N
* Office – Student – M:N

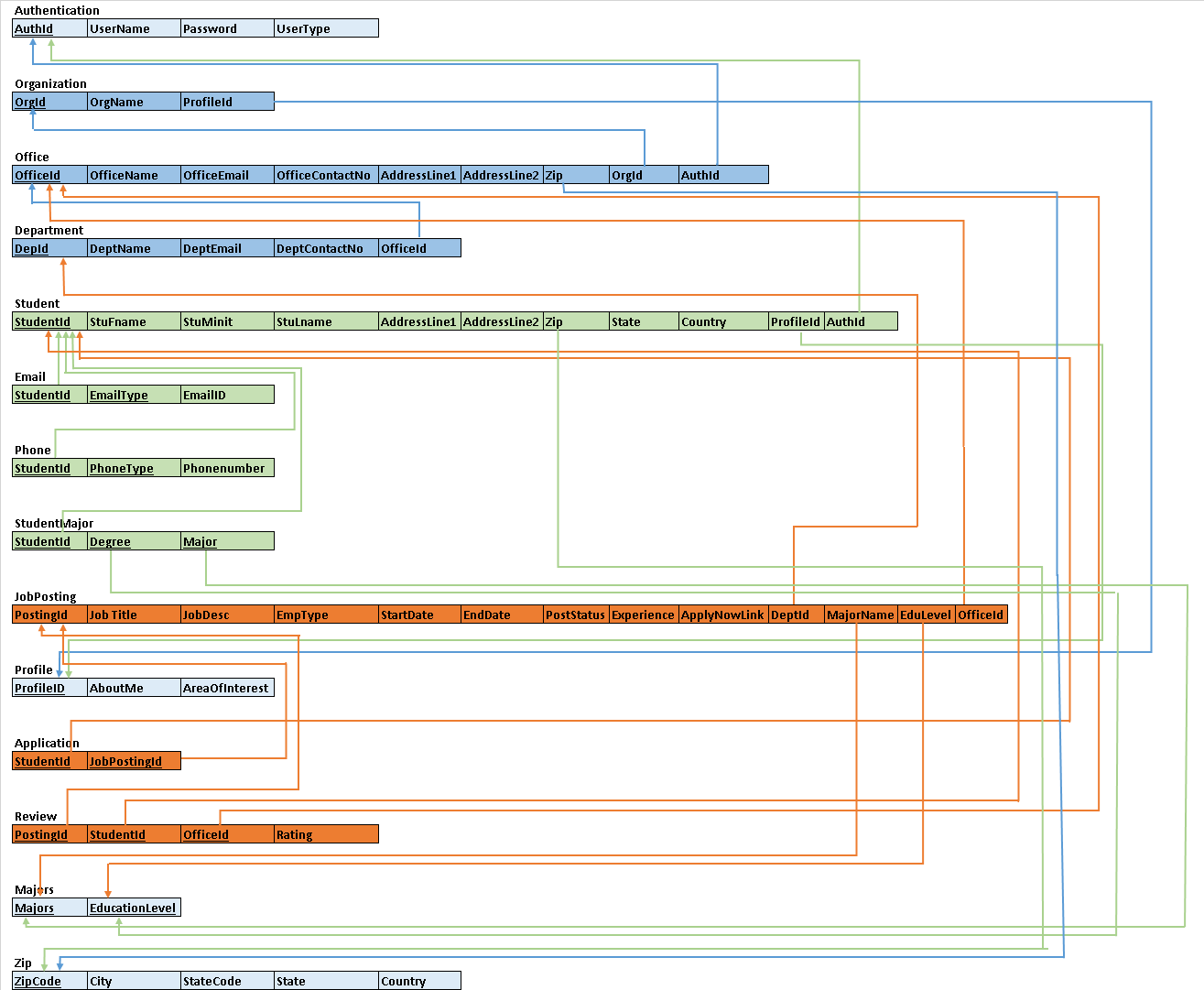
**Participation constraints:**

Participation constraints are mentioned in ER diagram using || lines

**Data Model:**



**Explanation of Data Model to SQL Tables:**

Below is the schema of the database created using Data Model.

|  |  |
| --- | --- |
|  | Tables common to both Organization and Students |
|  | Tables having information about Organization |
|  | Tables having information about Student |
|  | Tables having information about job posting |

Below is the explanation of the tables used

**Authentication:** This table stores the username and password of users. Details are filled in this table after HireIT Staff of the portal registers Students or Organization Office. Each Username and password stores two more values corresponding to it, AuthId and UserType.

AuthId : is used as surrogate key for username. For security reasons, this value is passed from one form to other form instead of actual username. Hence, session is maintained using AuthId

Usertype: This field hold “ORG” for Organization user and “STU” for Student user. It helps to show two different view of a single form based on the type of user.

After registration is completed Student or Organization Office can login to the portal using these details.

**Organization:** This table holds the basic Organization details and has foreign key of ProfileID referencing PROFILE table where the profile details are stored for the table.

**Office:** Organization can have multiple offices at multiple location. These office details are stored in this table.

\*\*Point to note here is that Office, instead of Organization, is having AuthId because we are assuming Office to login and perform actions on behalf of Organization

**Department:** Each Office can have multiple department and those details are stored in this table.

**Student:** This table has the basic information of a student which is entered while registering to the portal. This table does not have Email, Contact phone, Education details. These are mutivalued attributes and hence, are stored in a different table.

**Email:** This holds the multiple emails of a student with primary key as combination of StudentId and EmailType

**Phone:** This holds the multiple phone numbers of a student with primary key as combination of StudentId and PhoneType

**StudentMajor:** This holds the multiple Degree qualification of a student with primary key as combination of StudentId, Degree and Major

**JobPosting:**  This table enables Organization with the capability to post a job on the portal. Particular office of the organization need to login and create a job post for the available position. They need to select the department for which position is available. PostingId is the primary key along with OfficeId and DeptId as foreign key

**Profile:** This table holds the profile details which tells additional features of Organization as well as Student. Since both Organization and Student are having common fields for profile type information, we have generalized this property and have created a separate entity for the same.

**Application:** When a student applies for a job, an entry is logged into this table, so that Office can go and check details of all the students who applied for the job posting.

**Review:** This is enables a unique functionality of reviewing the job done by any candidate in form of rating which helps build the credibility of students. This feature can also be used by Office to hire students with a better rating. The ratings are provided for each job student completes. However, Offices are displayed with only the average rating of the student across all the jobs completed by him, ignoring the ones where students are not at all rated.

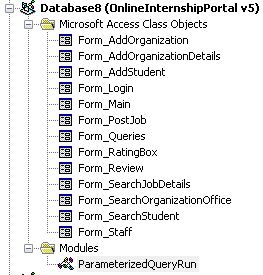
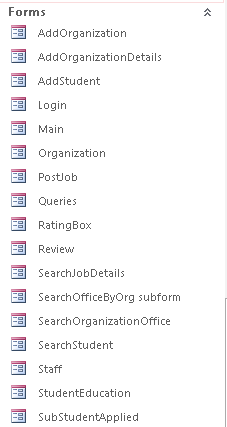
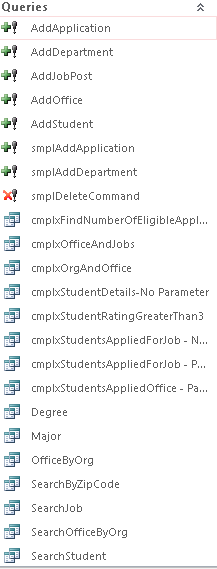
**Majors:** This is a repository ofall the education degree and majors offered across United states. Student registration and Job posting feature utilize this tables data to populate the data on the forms.

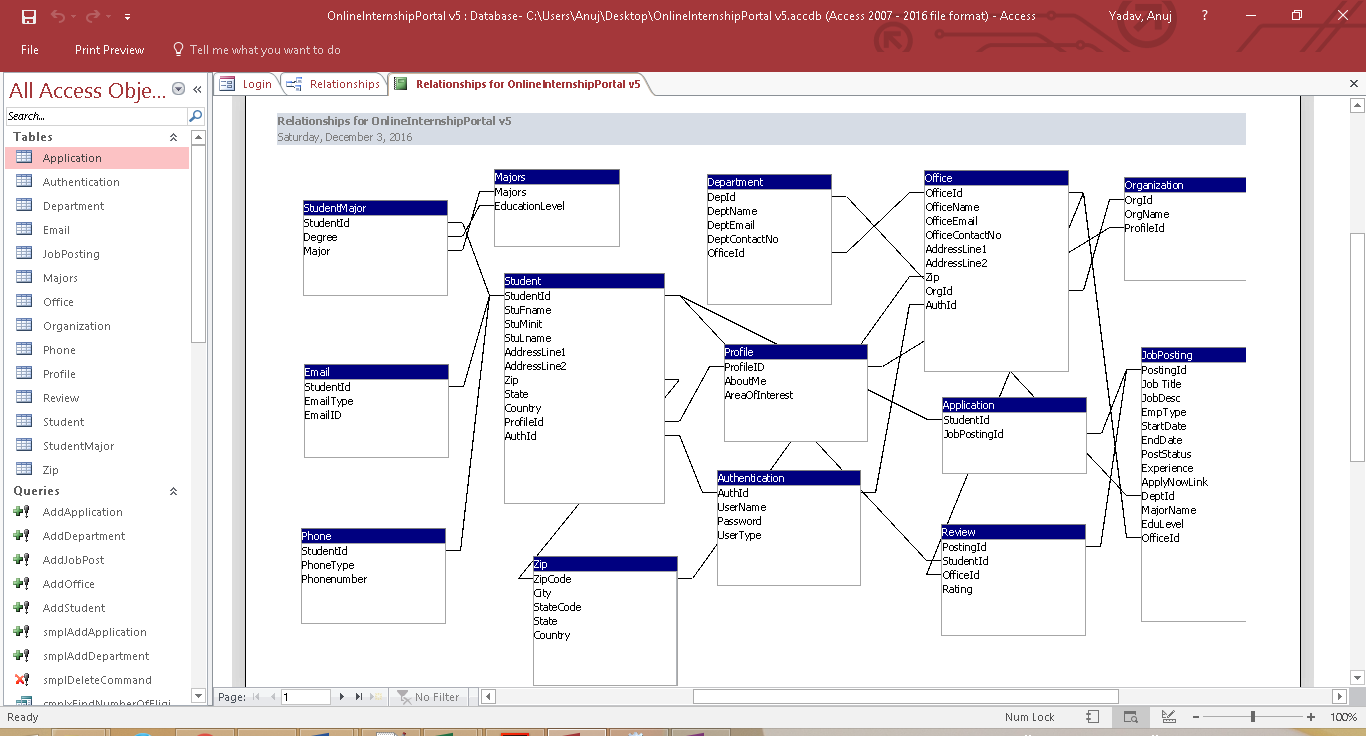
**Zip:** This is a repository ofall the ZipCodes and its details across United states. Student registration and Office registration feature utilize this tables data to populate the data on the forms.

**SQL Implementation:**

MS Access is used for the database as well as the application. We have used Tables, Queries, Forms and Reports all in one access project.

Custom forms are designed and have event procedure embedded to the action buttons and business logics are handled by VBA programming. Parameterized Queries are stored in Access to act as stored procedure. Parameters to these queries are passed either using VBA or dynamically from the values of form objects



Relationship between the table is also established in the Access itself. This ensures referential integrity between the tables

**Database Testing using Queries:**

**Simple Queries**

1. Record the application into database when a student applies for a job

Query -We have used Parameterized query in our implementation however, the same query is saved with name **‘smplAddApplication’** with default values

INSERT INTO Application VALUES ('8', '1');

2. Display the details of region based on Zipcode

Query –Although it’s a simple query, we have implemented it to auto populate the state and country in the form. It is saved as ‘**smplSearchByZipCode**’

SELECT Z.\*

FROM Zip AS Z

WHERE Z.ZipCode ='75252';

3. Mark all the job posting as Expired when the current date gets past end date of the Job

Query – We have used “O” as a flag to mark a job expired. Implementation uses parameterized query. However, for simplicity we have a copy of the query as **‘smpleUpdatePostStatus’** with default values

UPDATE JobPosting SET JobPosting.PostStatus = "O"

WHERE (JobPosting.EndDate)<Date();

4. Retrieve list of Active job posting for Jobs based on ‘Dallas or ‘Houston’

Query –This query is saved as **‘smplSearchJobBasedOnCity’** in access

SELECT JobPosting.[Job Title],

JobPosting.JobDesc,

JobPosting.EmpType,

JobPosting.StartDate,

JobPosting.EndDate,

JobPosting.Experience AS [Exp required],

Org.OrgName,

Z.City

FROM JobPosting, Office AS O, Organization AS Org, Zip as Z

WHERE JobPosting.PostStatus="N"

AND [O].[OrgId]=[Org].[OrgId]

AND [JobPosting].[OfficeId]=CStr([O].[OfficeId])

AND O.Zip = Z.ZipCode

AND (Z.City = 'DALLAS' or Z.City = 'HOUSTON') ;

5. Retrieve list of Active job posting for Jobs posted by “Operations” department and is eligible only for students with “Information System” major.

Query - This query is saved as **‘smplSearchJobBasedMajorAndDepartment’** in access

SELECT JobPosting.[Job Title],

JobPosting.JobDesc,

JobPosting.EmpType,

JobPosting.StartDate,

JobPosting.EndDate,

[JobPosting].[MajorName] AS [Major Required],

Org.OrgName,

D.DeptName

FROM JobPosting, Office AS O, Organization AS Org, Department AS D

WHERE JobPosting.PostStatus="N"

AND [O].[OrgId]=[Org].[OrgId]

AND [JobPosting].[OfficeId]=CStr([O].[OfficeId])

AND [O].[OfficeId] = D.OfficeID

AND D.DeptName LIKE '\*Operation\*'

AND [JobPosting].[MajorName] LIKE '\*Information System\*' ;

**Complex Queries**

1. Determine how many applicants for a job posting are qualified on the basis of their education major.

Query –We have tried to solve this problem using **GROUP BY** CLAUSE. This query is saved by the name **‘cmplxFindNumberOfEligibleApplications - No Parameter’**

SELECT A.JobPostingId as JobId,

JP.[Job Title],

JP.MajorName AS RequiredMajor,

JP.EduLevel AS EducationLevelRequired,

COUNT(A.StudentId) AS [Eligible Applicants]

FROM Application AS A, JobPosting AS JP, Student AS S

WHERE A.JobPostingId = JP.PostingId

and cstr(S.StudentId) = A.StudentId

and JP.MajorName in (Select SM.Major from StudentMajor as SM

where S.StudentId = SM.StudentId)

GROUP BY A.JobPostingId, JP.[Job Title], JP.EduLevel, JP.MajorName;

2. Not all Organization Offices of an Organization post a Job on the portal. List down all the offices registered on the portal along with job postings (if any) done by them

Query – This is a complex query because it cannot be achieved without using Outer Join. We have used **LEFT JOIN** to achieve the same. This query is saved by the name **‘cmplxOfficeAndJobs’**

SELECT Office.OfficeName,

JobPosting.[Job Title]

FROM Office LEFT JOIN JobPosting ON cstr(Office.OfficeId) = JobPosting.OfficeId;

3. Display details of students including email address, contact number, educational major.

Query - The details of Students is designed to have separate table for multivalued attribute, keeping in mind to have minimum null values. For example, we have identified Email address, Contact number and Education degree as multivalued attribute and hence have designed separate table for them. However, getting all the details in single view using simple query generates multiple rows for same students. We have employed **NESTED QUERIES** to display the multiple values of an attribute in separate columns instead of separate row. This query is saved by the name **‘cmplxStudentDetails-No Parameter’**

SELECT Student.StudentID,

Student.StuFname,

Student.StuMinit,

Student.StuLname,

(Select Email.EmailID from Email where Student.StudentId = Email.StudentId and Email.EmailType = 'Primary') AS PrimaryEmail,

(Select Email.EmailID from Email where Student.StudentId = Email.StudentId and Email.EmailType = 'Secondary') AS SecondaryEmail,

(Select Phone.Phonenumber from Phone where Student.StudentId = Phone.StudentId and Phone.PhoneType = 'Mobile') AS Mobile,

(Select Phone.Phonenumber from Phone where Student.StudentId = Phone.StudentId and Phone.PhoneType = 'Home') AS Home,

(Select Phone.Phonenumber from Phone where Student.StudentId = Phone.StudentId and Phone.PhoneType = 'Work') AS [Work],

Student.AddressLine1,

Student.AddressLine2,

Student.Zip, Profile.AboutMe AS About,

Profile.AreaOfInterest AS ['Interested in'],

Profile.ProfileId,

Student.AuthId

FROM Student, Profile

WHERE Student.ProfileID = Profile.ProfileID;

4. List details of students applied for a particular job so that Office can contact them and review their profile

Query - In this case we are searching for a Job with post id = 2 and have used **INNER JOIN** to join multiple tables. This query is saved by the name **‘cmplxStudentsAppliedForJob - No Parameter’**

SELECT JobPosting.PostingId,

Student.StuFname,

Student.StuMinit,

Student.StuLname,

(Select Email.EmailID from Email where Student.StudentId = Email.StudentId and Email.EmailType = 'Primary') AS PrimaryEmail,

(Select Email.EmailID from Email where Student.StudentId = Email.StudentId and Email.EmailType = 'Secondary') AS SecondaryEmail,

(Select Phone.Phonenumber from Phone where Student.StudentId = Phone.StudentId and Phone.PhoneType = 'Mobile') AS Mobile,

(Select Phone.Phonenumber from Phone where Student.StudentId = Phone.StudentId and Phone.PhoneType = 'Home') AS Home,

(Select Phone.Phonenumber from Phone where Student.StudentId = Phone.StudentId and Phone.PhoneType = 'Work') AS [Work],

Student.AddressLine1,

Student.AddressLine2,

Student.Zip,

Profile.AboutMe AS About,

Profile.AreaOfInterest AS [Interested in]

FROM ((JobPosting INNER JOIN Application ON JobPosting.PostingId =cstr(Application.JobPostingId))

INNER JOIN Student ON Application.StudentId = cstr(Student.StudentId))

INNER JOIN Profile ON Student.ProfileId = Profile.ProfileID

WHERE JobPosting.PostingId = '2';

5. List all the students having average rating provided by employer greater than 3 in descending order

Query – We have used **HAVING** Clause to achieve this. This query is saved by the name **‘cmplxStudentRatingGreaterThan3’**

SELECT S.StudentID,

S.StuFname & " " & S.StuMinit & " " & S.StuLname AS StudentName,

IIf(Avg(R.Rating) Is Null,0, format(Avg(R.Rating) , "#,##0.0")) AS [Average Rating]

FROM Review AS R RIGHT JOIN Student AS S ON R.StudentID = S.StudentID

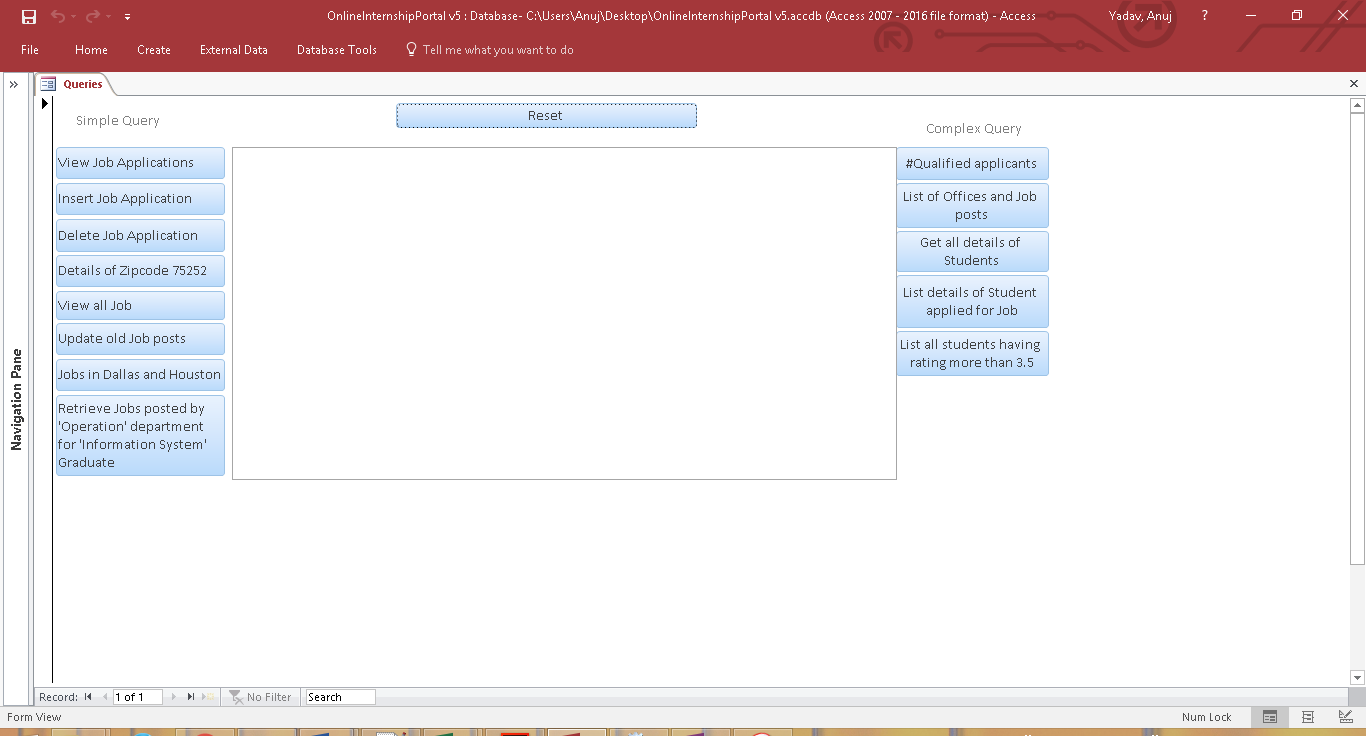
WHERE R.Rating is not null

GROUP BY S.StudentID, S.StuFname, S.StuMinit, S.StuLname

HAVING Avg(R.Rating) > 3

ORDER BY Avg(R.Rating) DESC;

These queries can also be executed from a custom form saved with name **“Queries”.** This is enabled for Staff to monitor database



**Conclusion:**

No of entities

Generalization-Specialization

No of Data Model tables

Implemented using MS Access 2016

Pages/Forms implemented – No, types (Login, search, update, retrieve etc.)

Complex queries, Simple queries used in front end forms for various functionalities