Course Selection Database

Project Report

Marie Hasegawa

Alexei Harris

Cole Young

Florida Polytechnic University

November 2020

Table of Contents

1 Project Overview

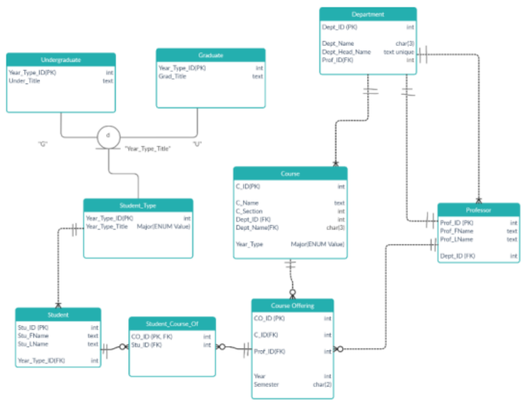
The goal of this project was to use two separate databases a relational PostgreSQL database and a nonrelational MongoDB database to create a working database system that could easily be implemented into a Course Registry online application for students attending higher-level education.

The PostgreSQL database act as a collection of relational tables that is used to organize data regarding a student's academic information that would determine the student's eligibility to a certain course that correlate to their degree and major.

The MongoDB database act as a nonrelational, document-driven database that stores the data in collections and documents that correlate to the course registration by indexing pending course registrations, pending course schedules, and recorded conversations between students and advisors concerning which courses would suit the students' need.

Commented [HM1]:

2 ERM (Entity Relationship Model) Diagram



2.1 Course Selection ERM Diagram

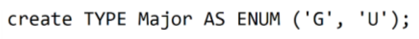
The Entity-Relationship Model, shown above, was used to develop the Course Selection PostgreSQL database, which will be further explained in the next section. The table, “Student_Type,” act as a subtype discriminator that determines the student as being a Graduate or Undergraduate, which are correlated to the subtype tables, “Undergraduate” and “Graduate.” The table, “Student_Course_Of ” act as a mediator between “Student” and “Course_Offering,” so that the variables “CO_ID” and “Stu_ID” can be contained in the same table. This has also led to having strong relationship between tables “Student,” “Student_Course_Of,” and “Course_Offering" since the “CO_ID” act as the primary key on the table, “Student_Course_Of.”

The tables “Student_Type” and “Student” have a weak, one to many relationship, since one “Student_Type” and can have many “Students” but a “Student” can only have one “Student_Type.” The tables “Course_Offering” and “Course” have a weak, many to one relationship, since a “Course” can have many “Course Offerings,” but a “Course” can have only one “Course_Offering.” The “Department” and “Course” tables have a weak one to many relationships, since a “Department” can have many “Courses,” but a “Course” can have only one “Department.” The tables, “Course_Offering” and “Professor” have a weak, many to one relationship, since a “Professor” can have many “Course_Offerings” but one “Course_Offering” can have only one “Professor.” The 1st relationship between tables “Department” and “Professor” show a weak, one-to-many relationship, since a “Department” can have many “Professors”, but one “Professor” can only have one “Department”. The 2nd relationship between tables “Department” and “Professor” show a weak, one-to-one relationship, since one

“Department” can only have one Head-of-Department and a Head-of-Department can only be head of one “Department”.

3 PostgreSQL Database

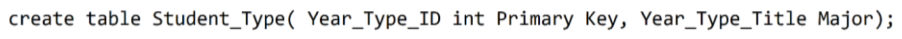
PostgreSQL is a relational database that corresponds to the ERM diagram shown in the previous section. In other words, the foreign keys, subtype discriminator, and the weak and strong relationships, and the one to one and many to many relationships will play a crucial role in the PostgreSQL portion of the “Course Selection” project. The database is used as a collection of relational tables that can be used to collect data about a student’s academic information which would determine the student’s eligibility to a course and if they can register in future semesters.



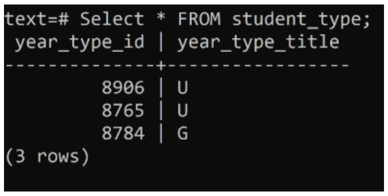
```
create TYPE Major AS ENUM ('G', 'U');
```

3.1 Major Enumerated Type

The code above creates an enumerated type called “Major” which plays as an identifier for Graduates (G) and Undergraduates (U). This enumerated type is used in the variable “Year_Type_Title,” which identifies a student being either a Graduate or an Undergraduate under the table “Student_Type.” In the table, “Course,” it has a column, “Year_Type,” that uses the enumerated type, “Major,” as an identifier of which Course is for Graduate students or Undergraduate Students.



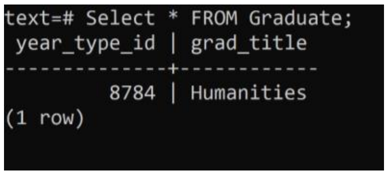
```
create table Student_Type( Year_Type_ID int Primary Key, Year_Type_Title Major);
```


```
text=# Select * FROM student_type;
 year_type_id | year_type_title
--------------+----------------
         8906 | U
         8765 | U
         8784 | G
(3 rows)
```

3.2 Student_Type Table

The table, “Student_Type,” acts as subtype discriminator that subtypes students as either a Graduate or Undergraduate, which correlates to the tables shown below, “Graduate” and “Undergraduate,” that act as subtype tables. The “Student_Type” table is a collection of rows that contain the information of a student’s “Year_Type_ID,” which acts as a primary key and ID number, and “Year_Type_Title", which is a “Major” enumerated type variable that identifies the student being a Graduate or an Undergraduate with character initials: “G” and “U.”

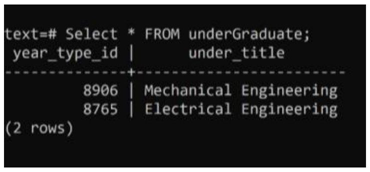
```
create table Graduate(Year_Type_ID int Primary Key REFERENCES Student_Type,
Grad_Title text);
```


```
text=# Select * FROM Graduate;
 year_type_id | grad_title
--------------+-----------
         8784 | Humanities
(1 row)
```

3.3 Graduate Table

The “Graduate” table is a subtype for the table “Student_Type,” which stores information regarding Graduate students’ major. It contains the Graduate’s “Year_Type_ID” and the string variable, “Grad_Title,” which acts as the Student’s major.

```
create table Undergraduate(Year_Type_ID int Primary Key REFERENCES Student_Type,
Under_Title text);
```



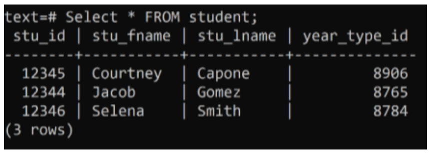
```
text=# Select * FROM underGraduate;
 year_type_id |      under_title
--------------+------------------------
         8906 | Mechanical Engineering
         8765 | Electrical Engineering
(2 rows)
```

3.4 Undergraduate Table

The “Undergraduate” table is a subtype for the table “Student_Type,” which stores information regarding Undergraduate students’ major. It contains the Undergraduate’s “Year_Type_ID” and the string variable, “Grad_Title,” which acts as the Student’s major.

```
create table Student( Stu_ID int Primary Key, Stu_FName text, Stu_LName text);
```

```
ALTER TABLE public.Student ADD COLUMN Year_Type_ID int,
ADD CONSTRAINT fk_test FOREIGN KEY (Year_Type_ID) REFERENCES Student_Type (Year_Type_ID);
```



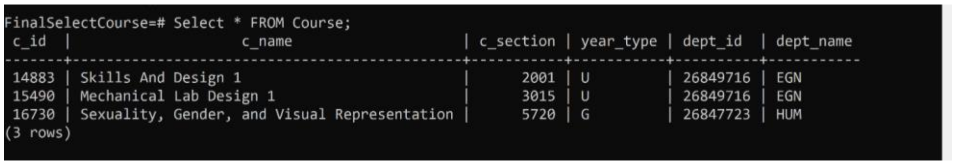
```
text=# Select * FROM student;
 stu_id | stu_fname | stu_lname | year_type_id
--------+-----------+-----------+--------------
  12345 | Courtney  | Capone    |         8906
  12344 | Jacob     | Gomez     |         8765
  12346 | Selena    | Smith     |         8784
(3 rows)
```

3.5 Student Table

The “Student” table stores data regarding student information. It contains the students “Stu_Id” which is the primary key of the table and uniquely identifies each student with an ID number. “Stu_Fname,” and “Stu_Lname” are the string columns of the first and last name of the student. The variable, “Year_Type_Id” is a foreign key from the table “Year_Type.”

```
create table Course( C_ID int Primary Key, C_Name text, C_Section int, Year_Type
Major);
```

```
ALTER TABLE public.Course ADD COLUMN Dept_ID int,
ADD CONSTRAINT fk_test3 FOREIGN KEY (Dept_ID) REFERENCES Department(Dept_ID);

ALTER TABLE public.Course ADD COLUMN Dept_Name text,
ADD CONSTRAINT fk_test4 FOREIGN KEY (Dept_Name) REFERENCES Department(Dept_Name);
```



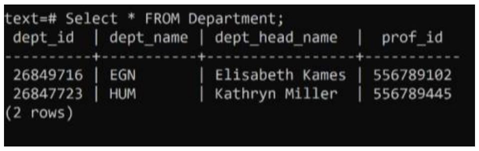
```
FinalSelectCourse=# Select * FROM Course;
 c_id  |                   c_name                    | c_section | year_type | dept_id  | dept_name
-------+---------------------------------------------+-----------+-----------+----------+----------
 14883 | Skills And Design 1                         |      2001 | U         | 26849716 | EGN
 15490 | Mechanical Lab Design 1                     |      3015 | U         | 26849716 | EGN
 16730 | Sexuality, Gender, and Visual Representation |      5720 | G         | 26847723 | HUM
(3 rows)
```

3.6 Course Table

The “Course Table” stores data regarding a Course’s information with 6 columns: “C_ID”, “C_Name”, “C_Section”, “Year_Type”, “Dept_ID”, and “Dept_Name”. The variables “C_ID”, “C_Name”, and “C_Section” are the Course’s id number, name, and section number. The variable “Year_Type” is an enumerated type, “Major”, which means that it identifies the “Courses” as either “Undergraduate” or “Graduate”. The “Dept_ID” and the “Dept_Name” are foreign keys from the table Department, which determines the Course’s “Department” ID and name.

```
create table Department( Dept_ID int Primary Key, Dept_Name text unique,
Dept_Head_Name text);
```

```
ALTER TABLE public.Department ADD COLUMN Prof_ID int,
ADD CONSTRAINT fk_test6 FOREIGN KEY (Prof_ID) REFERENCES Professor (Prof_ID);
```



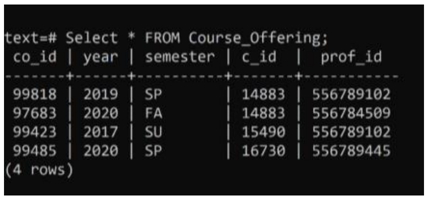
```
text=# Select * FROM Department;
 dept_id  | dept_name | dept_head_name  |  prof_id
----------+-----------+-----------------+-----------
 26849716 | EGN       | Elisabeth Kames | 556789102
 26847723 | HUM       | Kathryn Miller  | 556789445
(2 rows)
```

3.7 Department Table

The “Department” table contains information regarding departments. The “Dept_ID” and “Dept_Name” attributes are the ID number and name of the department. The “Dept_Head_Name” attribute has a string value of the Head-of-Department's full name. The “Prof_ID” is a foreign key from table “Professor” which refers to the Head-of-Department's “Professor” ID.

```
create table Course_Offering( CO_ID int Primary Key, Year int, Semester char(2));
```

```
ALTER TABLE public.Course_Offering ADD COLUMN C_ID int,
ADD CONSTRAINT fk_test1 FOREIGN KEY (C_ID) REFERENCES Course (C_ID);

ALTER TABLE public.Course_Offering ADD COLUMN Prof_ID int,
ADD CONSTRAINT fk_test2 FOREIGN KEY (Prof_ID) REFERENCES Professor (Prof_ID);
```



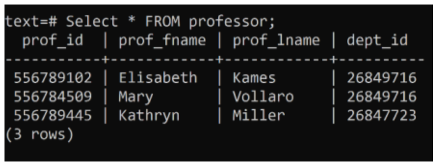
```
text=# Select * FROM Course_Offering;
 co_id | year | semester | c_id  |  prof_id
-------+------+----------+-------+-----------
 99818 | 2019 | SP       | 14883 | 556789102
 97683 | 2020 | FA       | 14883 | 556784509
 99423 | 2017 | SU       | 15490 | 556789102
 99485 | 2020 | SP       | 16730 | 556789445
(4 rows)
```

3.8 Course_Offering Table

The table, “Course_Offering Table”, is a personal collection of information regarding a “Course” offered to a particular Student for future semesters in the Course Registry. The attributes, “Co_ID”, “Year”, and “Semester” is an ID number, year, and semester of the “Course” offer. The variables, “C_ID” and “Prof_ID”, are foreign keys from the tables “Course” and “Professor.”

```
create table Professor( Prof_ID int Primary Key, Prof_FName text, Prof_LName text);
```

```
ALTER TABLE public.Professor ADD COLUMN Dept_ID int,
ADD CONSTRAINT fk_test12 FOREIGN KEY (Dept_ID) REFERENCES Department(Dept_ID);
```

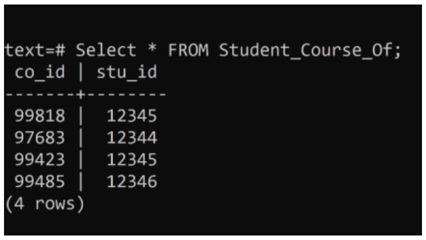


```
text=# Select * FROM professor;
  prof_id  | prof_fname | prof_lname | dept_id
-----------+------------+------------+----------
 556789102 | Elisabeth  | Kames      | 26849716
 556784509 | Mary       | Vollaro    | 26849716
 556789445 | Kathryn    | Miller     | 26847723
(3 rows)
```

3.9 Professor Table

The table, “Professor”, has a collection of rows that hold academic and personal information of the “Professors”. The attributes, “Prof_ID”, “Prof_FName”, and “Prof_LName” indicates the Professor’s ID number, first name, and last name. The column, “Dept_ID”, is a foreign key from the table, “Department”, which acts as the Professor’s “Department” ID number.

```
create table Student_Course_Of( CO_ID int Primary Key REFERENCES Course_Offering);

ALTER TABLE public.Student_Course_Of ADD COLUMN Stu_ID int,
ADD CONSTRAINT fk_test7 FOREIGN KEY (Stu_ID) REFERENCES Student(Stu_ID);
```



```
text=# Select * FROM Student_Course_Of;
 co_id | stu_id
-------+--------
 99818 |  12345
 97683 |  12344
 99423 |  12345
 99485 |  12346
(4 rows)
```

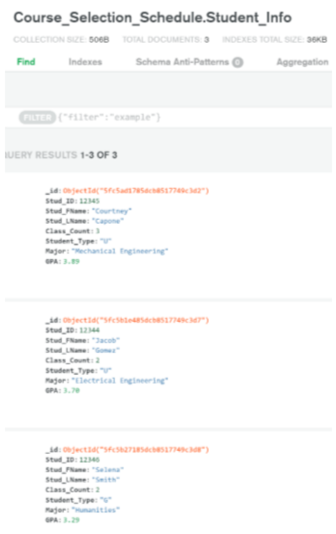
3.10 Student_Course_Of Table

The table, “Student_Course_Of”, act as a mediator between “Student” and “Course_Offering,” so that the variables “CO_ID” and “Stu_ID” can be contained in the same table. The variable “CO_ID” act as a primary key, but it is also a foreign key from “Course_Offering”, so the tables “Course_Offering” and “Student_Course_Of” has a strong relationship.

4 MongoDB Database

MongoDB is a nonrelational, document storing database that we used to store the information that we had relating to the individual aspects of our database. Our MongoDB Database was made to store documents and collections regarding the course registration with aspects of pending course registrations, pending course schedules, and recorded conversations between students and advisors concerning which courses would suit the students' need. This type of database can be used for the pending schedule webpage in a Course registration website and create a private online chat room between students and advisors where it can collect their conversations regarding course selection. This is especially more useful as a MongoDB database, since indexing conversations through a relational database would be impractical since all conversations would be unique.

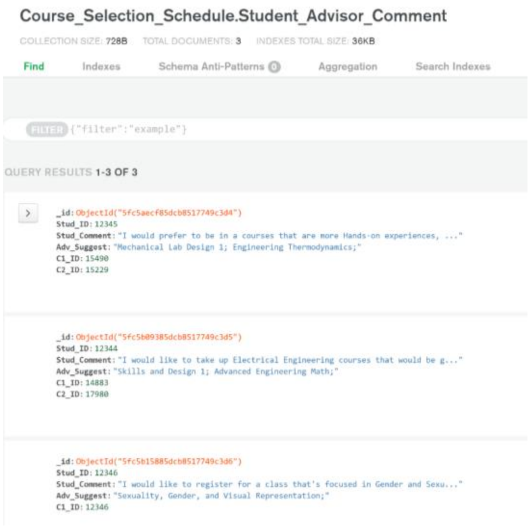
Commented [HM2]: Follow Up: registraion



```
_id: ObjectId("5fc5ad1785dcb8517749c3d2")
Stud_ID: 12345
Stud_FName: "Courtney"
Stud_LName: "Capone"
Class_Count: 3
Student_Type: "U"
Major: "Mechanical Engineering"
GPA: 3.89

_id: ObjectId("5fc5b1e485dcb8517749c3d7")
Stud_ID: 12344
Stud_FName: "Jacob"
Stud_LName: "Gomez"
Class_Count: 2
Student_Type: "U"
Major: "Electrical Engineering"
GPA: 3.70

_id: ObjectId("5fc5b27185dcb8517749c3d8")
Stud_ID: 12346
Stud_FName: "Selena"
Stud_LName: "Smith"
Class_Count: 2
Student_Type: "G"
Major: "Humanities"
GPA: 3.29
```

4.1 Student_Info Collection

The “Student_Info” collection has documents regarding a Student’s academic information to be used for Course Registration. The documents have 8 attributes: “_id”, “Stud_ID”, “Stud_FName”, “Stud_LName”, “Class_Count”, “Student_Type”, “Major”, and “GPA”. The attribute, “_id,” acts as the primary key or id key since every document contains one in MongoDB. The “Stud_ID” attribute is the id number of a student, while “Stu_FName” and “Stu_LName” are the first and last names of a student. The attribute, “Class_Count,” act as an integer variable that counts the number of a student’s pending and completely registered courses. The attribute, “Major,” act as a string variable that identifies a student’s major, while the “GPA” attribute is a decimal variable that clarifies a student’s GPA.

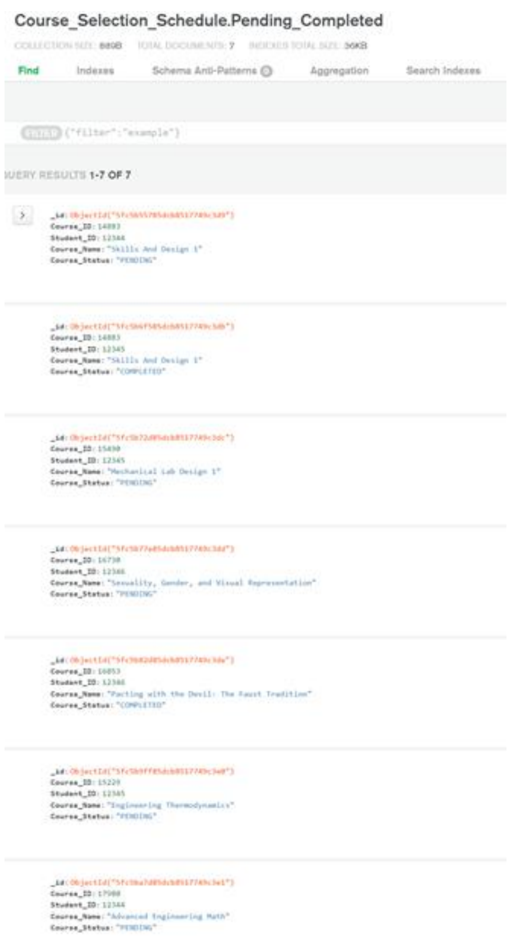

Course_Selection_Schedule.Student_Advisor_Comment
COLLECTION SIZE: 728B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
FILTER
QUERY RESULTS 1-3 OF 3

```
{"filter":"example"}
```

```
_id: ObjectId("5fc5aecf85dcb8517749c3d4")
Stud_ID: 12345
Stud_Comment: "I would prefer to be in a courses that are more Hands-on experiences, ..."
Adv_Suggest: "Mechanical Lab Design 1; Engineering Thermodynamics;"
C1_ID: 15490
C2_ID: 15229

_id: ObjectId("5fc5b09385dcb8517749c3d5")
Stud_ID: 12344
Stud_Comment: "I would like to take up Electrical Engineering courses that would be g..."
Adv_Suggest: "Skills and Design 1; Advanced Engineering Math;"
C1_ID: 14883
C2_ID: 17980

_id: ObjectId("5fc5b15885dcb8517749c3d6")
Stud_ID: 12346
Stud_Comment: "I would like to register for a class that's focused in Gender and Sexu..."
Adv_Suggest: "Sexuality, Gender, and Visual Representation;"
C1_ID: 12346
```

4.2 Student_Advisor_Comment Collection

The “Student_Advisor_Comment” Collection is a table of conversations between students and advisors. A document from this table has 5 attributes that contain details about the conversation: “_id”, “Stud_ID”, “Adv_Suggest”, and “C#_ID”. The “Stud_Comment” attribute is the string variable that contains the student's statement of what preference they want for the course, while the string attribute, “Adv_Suggest", acts as the Advisor's course suggestion to the student's preferences. The attribute group, “C#_ID,” is a collection of integer variables that identify suggested courses' ID, and the amount of “C#_ID” is dependent on the number of suggestions under the variable, “Adv_Suggest.”



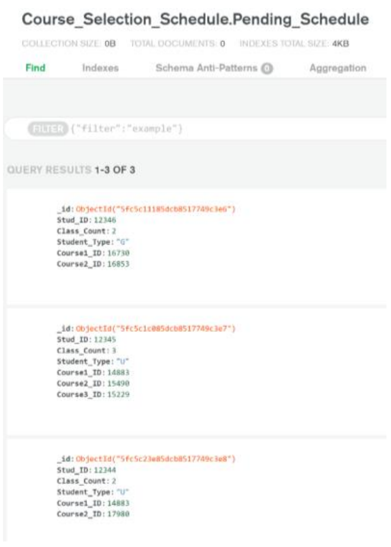
Course_Selection_Schedule.Pending_Completed

COLLECTION SIZE: 880B TOTAL DOCUMENTS: 7 INDEXES TOTAL SIZE: 36KB

```
{"filter":"example"}

QUERY RESULTS 1-7 OF 7

_id: ObjectId("5fc5b65785dcb8517749c3d9")
Course_ID: 14883
Student_ID: 12344
Course_Name: "Skills And Design 1"
Course_Status: "PENDING"

_id: ObjectId("5fc5b66f85dcb8517749c3db")
Course_ID: 14883
Student_ID: 12345
Course_Name: "Skills And Design 1"
Course_Status: "COMPLETED"

_id: ObjectId("5fc5b72d85dcb8517749c3dc")
Course_ID: 15430
Student_ID: 12345
Course_Name: "Mechanical Lab Design 1"
Course_Status: "PENDING"

_id: ObjectId("5fc5b77e85dcb8517749c3dd")
Course_ID: 16730
Student_ID: 12346
Course_Name: "Sexuality, Gender, and Visual Representation"
Course_Status: "PENDING"

_id: ObjectId("5fc5b82d85dcb8517749c3de")
Course_ID: 16853
Student_ID: 12346
Course_Name: "Pacting with the Devil: The Faust Tradition"
Course_Status: "COMPLETED"

_id: ObjectId("5fc5b9ff85dcb8517749c3e0")
Course_ID: 15229
Student_ID: 12345
Course_Name: "Engineering Thermodynamics"
Course_Status: "PENDING"

_id: ObjectId("5fc5ba7d85dcb8517749c3e1")
Course_ID: 17988
Student_ID: 12344
Course_Name: "Advanced Engineering Math"
Course_Status: "PENDING"
```

4.3 Pending_Completed Collection

The “Pending_Completed” Collection is a table of pending and completely registered courses of specific students. The variables, “Course_ID” and “Course_Name,” act as a course id and name of a registered course, while the “Student_ID” attribute act as the id number of the student who is registering for that specific course. The attribute, “Course_Status,” acts as the status of the registering course as either “PENDING” or “COMPLETED.”



```
{"filter":"example"}
```

```
_id: ObjectId("5fc5c11185dcb8517749c3e6")
Stud_ID: 12346
Class_Count: 2
Student_Type: "G"
Course1_ID: 16730
Course2_ID: 16853
```

```
_id: ObjectId("5fc5c1c085dcb8517749c3e7")
Stud_ID: 12345
Class_Count: 3
Student_Type: "U"
Course1_ID: 14883
Course2_ID: 15490
Course3_ID: 15229
```

```
_id: ObjectId("5fc5c23e85dcb8517749c3e8")
Stud_ID: 12344
Class_Count: 2
Student_Type: "U"
Course1_ID: 14883
Course2_ID: 17980
```

4.4 Pending_Schedule Collection

The “Pending_Status” Collection has documents that act as a student’s undetermined schedule of pending and completely registered courses for future semesters. The collection has 6 attributes: “_id”, “Stud_ID”, “Student_Type”, and “Course#_ID”. The attributes, “Stu_ID” and “Student_Type,” refers to the student’s id number and degree level (Undergraduate or Graduate). The “Class_Count” variable counts the number of pending and completely registered courses the student has applied for. The variable group, “Course#_ID,” is a list of variables that list all the id numbers of pending and completely registered courses.

5 Conclusion

The project has successfully implemented a Course Selection PostgreSQL database and a Scheduling Advice MongoDB database which could be used for a Course Registry online application for students attending higher-level education.

The PostgreSQL database organizes academic data about “Student”, “Course”, “Department”, and “Professor” information in order to help implement a theoretical Course Registry Application. It would do this by organizing the course registration data of a student in a relational manner.

The MongoDB database is document-driven that indexes collections and documents that focus on pending course registrations, pending course schedules, and recorded conversations between advisors and students regarding which courses suit the student’s preferences.

6 References

99 Unique Database Project Ideas for Final Year Students. (2020, September 29). Retrieved from https://instanteduhelp.com/unique-database-project-ideas/

Domínguez, C., & Jaime, A. (2010, June 16). Database design learning: A project-based approach organized through a course management system. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S0360131510001600

Vats, R (2020) "15 Exciting SQL Project Ideas & Topics for Beginners" UpGrad Blog Retrieved from: https://www.upgrad.com/blog/sql-project-ideas-topics-for-beginners/