Yi Lin

CMPT220

Professor Arias

6 April 2018

Project 2 Writeup

My database project utilizes three table: student data table, Marist data table, and location data table. The Student data table stores the academic, financial, and family information of the students. The Marist data table stores the requirements for the students to continue college. The location data table stores the students’ home town economy information. The java program will pull data from those table and use students’ data as independent variables and location data and Marist data as dependent variables to calculate the possibility for student to continue college. I will use academic performance and financial ability as the measurement for student continuity.

My determination to work on database project was inspire from the project that my supervisor and colleague are working in their course. Their project focus student retention, but, unlike my database project, they consider various factors that allow student to continue college academic career. They consider not only academic performance and financial ability, but also psychological factors that cause student to dropout, travel distance that cause student to transfer, social issues that student faced in current college, and opportunities that the college offer. After listen to their discussion, I want to do similar things for my database project.

As the first step after proposing the project, I began with constructing a database starting with student data table, Marist data table and then location data table. For accessibility and easiness, I decide to use Microsoft Access to construct the database. I fill student table with fictional student information. Then, I used data from Marist websites to construct Marist table and used data from census.gov to construct location table. After completing those tables in the database, I started to write java codes to build the connection to the databases. I searched online and found sample java codes that construct the Microsoft connection using a JDBC driver called ucannaccess. I follow the sample code and create my first prototype, but it prompted error when running. I went to my colleagues and professor for help and discovered that one of the reference libraries that the driver require was outdate. After I updated the library, the code run fine. I began to pull data from student table and measure it against Marist table. I create three difference classes for each table from extending a general database superclass to construct the connections to those table and pull data. During the process, syntax error on SQL queries in java codes and methods to pull the data have cause much delay. After pulling the data from the table, my next objective was to write the algorithm to predict the students’ GPA at the end of the semester and the possibility that students’ family will pay offer cost of attendance if there is any remaining. To predict the students’ GPA, I will use the class performance of the students and the number of the credit for that class. First, I will multiply the current grade of the class and the credit of the class for every class that student has registered. Second, I will sum the products and the divide the sum by the total amount of credit the student registered. Third, after dividing the sum, I add the result to the current GPA of the students and divide the result by two. If the student does not have a GPA, GPA equal to zero, then the result of dividing the sum will be the predict GPA of the student. After predicting the GPA of the student, I will check the GPA against the minimum GPA require for student to avoid academic probation. If the predicted GPA is less than the minimum, then the amount of GPA needed will be calculate by minus the require GPA by the predicted GPA. To calculate the students’ family’s financial ability to contribute to the cost of attending, I first research the average cost of living in the city where the student live. Then I use the data as filter to determine at what level of income the family will contribute to the cost of attendance. The filtered result will determine the possibility of family contribution for each income level and the total possibility of family contribution in that local area. Next, I find the difference between the filtered level of income and the cost of living. The difference will be the maximum amount that a family of that range of income can contributes. After finishing the predicting the GPA and calculate the family’s contribution, I will print them in as two table per student. Initially, I print the result as sentences rather than table. For better view of the data, I decide the print the data as two tables. The first table will have student’s information, total possibility of family contribution, and the amount of money and GPA student need to continue school. The second table will print the student’s family contribution and the possibility in according to each income level.

Utilizing both table, the user can have better understand the possibility for student to continue school in the end of the semester from the perspective in any point in the semester. The best timing to use this program is in the mid-semester when midterm grades are being post to student. Mid-semester have enough information that can be user to predict student’s performance at the end of the semester. The mid-semester also a good time for student to start making change to help them continue college. This program will allow the user to understand how the student’s need to continue college from academic and financial perspective. I wish by accomplish the project I will gain more experience and skills in write code in regarding java.

UML

|  |  |  |  |
| --- | --- | --- | --- |
| database | | | |
|  | | | |
| +Database()  +connect(): Connection | +close(Connection conn): void | +pullData(Connection conn): void | +pullData(Connection conn, String home): void |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MaristDB | | | | | | | |
| -tuition: long | -roomAndBoard:long | | -bookAndSupplies:long | | -personalMiscellaneous: long | | -minGPA: double |
| + MaristDB ()  + connect(): Connection | | + pullData(Connection conn): void  + getPersonalMiscellaneous(): long | | + getTuition(): long  + close(): void | | + getBookAndSupplies(): long  + getMinGPA(): double  + getRoomAndBoard(): long | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StudentDB | | | | |
| -ids: ArrayList<Integer>  -firstName: ArrayList<String>  -lastName: ArrayList<String>  -home: ArrayList<String> | | -gpa: ArrayList<Double>  -classPerform: ArrayList<Integer>  -studentStatus: ArrayList<String> | | -awardSch: ArrayList<Integer>  -thirdSch: ArrayList<Integer>  -famCont: ArrayList<Integer> |
| + StudentDB()  + connect(): Connection  + close(): void  + pullData(Connection conn): void  + getIds():ArrayList<Integer> | + getFirstName(): ArrayList<String>  + getLastName(): ArrayList<String>  + getGpa():ArrayList<Double>  + getClassPerform(): ArrayList<Integer>  + getAwardSch(): ArrayList<Integer> | | + getThirdSch(): ArrayList<Integer>  + getFamCont(): ArrayList<Integer>  + getHome(): ArrayList<String>  + getStudentStatus(): ArrayList<String> | |

|  |  |  |
| --- | --- | --- |
| LocationDB | | |
| -city: String  -state: String  -totalHousehold: int  -families: int  -marriedCouple: int  -nonfamily: int  -totalHouseholdMedianIncome: int  -familiesMedianIncome: int  -marriedCoupleMedianIncome: int  -nonfamilyMedianIncome: int  -totalHouseholdMeanIncome: int  -familiesMeanIncome: int  -marriedCoupleMeanIncome: int  -nonfamilyMeanIncome: int  -less10kHouseholdPer: double  -less10kFamiliesPer: double  -less10kMarriedCouplePer: double  -less10kNonfamilyPer: double | -less15kHouseholdPer: double  -less15kFamiliesPer: double  -less15kMarriedCouplePer: double  -less15kNonfamilyPer: double  -less25kHouseholdPer: double  -less25kFamiliesPer: double  -less25kMarriedCouplePer: double  -less25kNonfamilyPer: double  -less35kHouseholdPer: double  -less35kFamiliesPer: double  -less35kMarriedCouplePer: double  -less35kNonfamilyPer: double  -less50kHouseholdPer: double  -less50kFamiliesPer: double  -less50kMarriedCouplePer: double  -less50kNonfamilyPer: double  -less75kHouseholdPer: double  -less75kFamiliesPer: double | -less75kMarriedCouplePer: double  -less75kNonfamilyPer: double  -less100kHouseholdPer: double  -less100kFamiliesPer: double  -less100kMarriedCouplePer: double  -less100kNonfamilyPer: double  -less150kHouseholdPer: double  -less150kFamiliesPer: double  -less150kMarriedCouplePer: double  -less150kNonfamilyPer: double  -less200kHouseholdPer: double  -less200kFamiliesPer: double  -less200kMarriedCouplePer: double  -less200kNonfamilyPer: double  -more200kHouseholdPer: double  -more200kFamiliesPer: double  -more200kMarriedCouplePer: double  -more200kNonfamilyPer: double |
| +LocationDB ()  +connect(): Connection  +close(): void  +pullData(Connection conn): void  +getCity(): String  +getState: String  +getTotalHousehold: int  +getFamilies: int  +getMarriedCouple: int  +getNonfamily: int  + getTotalHouseholdMedianIncome: int  +getFamiliesMedianIncome: int  +getMarriedCoupleMedianIncome: int  +getNonfamilyMedianIncome: int  +getTotalHouseholdMeanIncome: int  +getFamiliesMeanIncome: int  +getMarriedCoupleMeanIncome: int | +getNonfamilyMeanIncome: int  +getLess10kFamiliesPer: double  +getLess10kHouseholdPer: double  +getLess10kMarriedCouplePer: double  +getLess10kNonfamilyPer: double  +getLess15kHouseholdPer: double  +getLess15kFamiliesPer: double  +getLess15kMarriedCouplePer: double  + getLess15kNonfamilyPer: double  +getLess25kHouseholdPer: double  +getLess25kFamiliesPer: double  +getLess25kMarriedCouplePer: double  +getLess25kNonfamilyPer: double  +getLess35kHouseholdPer: double  +getLess35kFamiliesPer: double  +getLess35kMarriedCouplePer: double  +getLess35kNonfamilyPer: double  +getLess50kHouseholdPer: double  +getLess50kFamiliesPer: double  +getLess50kMarriedCouplePer: double  +getLess50kNonfamilyPer: double | +getLess75kHouseholdPer: double  +getLess75kFamiliesPer: double  +getLess75kMarriedCouplePer: double  +getLess75kNonfamilyPer: double  +getLess100kHouseholdPer: double  +getLess100kFamiliesPer: double  +getLess100kMarriedCouplePer: double  +getLess100kNonfamilyPer: double  +getLess150kHouseholdPer: double  +getLess150kFamiliesPer: double  +getLess150kMarriedCouplePer: double  +getLess150kNonfamilyPer: double  +getLess200kHouseholdPer: double  +getLess200kFamiliesPer: double  +getLess200kMarriedCouplePer: double  +getLess200kNonfamilyPer: double  +getMore200kHouseholdPer: double  +getMore200kFamiliesPer: double  +getMore200kMarriedCouplePer: double  +getMore200kNonfamilyPer: double |

Reference

SJ. “Java JDBC: An Example to Connect MS Access Database.” BenchResources.Net, 24 Apr. 2017, www.benchresources.net/jdbc-msaccess-database-connection-steps/. Accessed 2 April 2018

Saleem, Aamir, director. Java Program to Access the MS Access Database with Ucanaccess Library with Java 8 or Above. Youtube, 2 Dec. 2016, www.youtube.com/watch?v=Zb1AWUroicM. Accessed 2 April 2018

DePersio, Greg. “How Much Money Do You Need to Live in Miami?” Investopedia, Investopedia, 8 Sept. 2015, www.investopedia.com/articles/personal-finance/090815/how-much-money-do-you-need-live-miami.asp. Accessed 28 March 2018

DePersio, Greg. “How Much Money Do You Need to Live in NYC?” Investopedia, Investopedia, 9 Mar. 2018, www.investopedia.com/articles/personal-finance/092415/how-much-money-do-you-need-live-nyc.asp. Accessed 28 March 2018