LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

CC6001NI Advanced Database System Development

40% Individual Coursework

Student Name: Ishan Chemjong

London Met ID: 19031002

College ID: NP01CP4A190293

Assignment Due Date: 23 March 2022

Assignment Submission Date: 22 March 2022

Word Count: 11319

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Introduction 1

Textual Analysis 3

ERD from Case Study 7

Normalization 8

- Normalization of Figure 1 10
- Normalization of Figure 2 14

Integration and Assumption 18

Final ERD 20

Data Dictionary 21

Script 33

- Generation of Database 33
- Create Statements 34

Insert Statements 47

Select Statements 60

Forms 72

- Dashboard or Home Page 72
- Simple Forms 74
- Complex Forms and SQL Queries 79
  - Complex Forms 79
  - SQL Queries 82

Testing 84

User Manual 121

Further Discussion 137

Conclusion 138

References 139

Table of Figures

Figure 1: Textual Analysis of College and Departments 3
Figure 2: Textual Analysis of Department and Examinations 3
Figure 3: Textual Analysis of Department and Assignment 4
Figure 4: Textual Analysis of Department and Result 4
Figure 5: Textual Analysis of Department and Fee 4
Figure 6: Textual Analysis of Department and Attendance 5
Figure 7: Textual Analysis of Student and Module 5
Figure 8: Textual Analysis of Teacher and Module 6
Figure 9: Textual Analysis of Person, Student and Techer 6
Figure 10: Initial Raw ERD from Case Study 7
Figure 11: Figure 1 for normalization 9
Figure 12: Figure 2 for normalization 13
Figure 13: Final ER diagram of the database 19
Figure 14: Creating a new user in oracle, Berkeley_College 32
Figure 15: Granting all the privileges to Berkeley_College user 32
Figure 16: Executing the DDL script for creating tables in SQL Developer 42
Figure 17: Tables created after executing DDL script in SQL developer..... 43
Figure 18: Altering tables with foreign keys in SQL developer 44
Figure 19 Tables altered with foreign keys in SQL developer..... 45
Figure 20: Inserting values in Address Table 46
Figure 21: Inserting values in Person Table..... 47
Figure 22: Inserting values in Person_Address_info Table 48
Figure 23: Inserting values in Student Table 49
Figure 24: Inserting values in Teacher Table 50
Figure 25: Inserting values in Departments Table 51
Figure 26: Inserting values in Module Table 52
Figure 27: Inserting values in Teacher_Module_info Table 53
Figure 28: Inserting values in Assignment Table 54
Figure 29: Inserting values in Student_Attendance_info Table 55
Figure 30: Inserting values in Student_Fee_Payment Table 56
Figure 31: Inserting values in Module_Student Table..... 57
Figure 32: Inserting values in Result_info Table 58
Figure 33: Select statement in Address Table 59
Figure 34: Select statement in Person Table 60
Figure 35: Select statement in Assignment Table 61
Figure 36: Select statement in Departments Table 61
Figure 37: Select statement in Module Table 62
Figure 38: Select statement in Person_Address_info Table 63
Figure 39: Select statement in Result_info Table..... 64
Figure 40: Select statement in Student Table 65
Figure 41: Select statement in Student_Attendance_info Table 66
Figure 42: Select statement in Student_Fee_Payment Table 67
Figure 43: Select statement in Teacher Table 68
Figure 44: Select statement in Teacher_Module_info Table 69
Figure 45: Select statement in Module_Student Table 70
Figure 46: Dashboard Page of Berkeley College 71
Figure 47: Simple Form of Student Table 72

Figure 48: Simple Form of Department Table 73
Figure 49: Simple Form of Teacher Table 74
Figure 50: Simple Form of Address Table 75
Figure 51: Simple Form of Module Table 76
Figure 52: Complex Form for Teacher Module Mapping 77
Figure 53: Complex Form for Student Fee Payment Details 78
Figure 54: Complex Form for Student Assignment Details..... 79
Figure 55: Reading data from Student Table 82
Figure 56: Student Table 83
Figure 57: Inserting New Student Record 84
Figure 58: Inserted New Student Record shown in table 84
Figure 59: Student Table 85
Figure 60: Updating a Student Record 86
Figure 61: Updated Student Record shown in table 86
Figure 62: Student Table before deleting a Student Record 87
Figure 63: Deleting a Student Record 88
Figure 64: Student Table after deleting a Student Record 88
Figure 65: Reading data from Department Table 89
Figure 66: Department Table 90
Figure 68: Inserting New Department Record shown in table 91
Figure 69: Department Table 92
Figure 70: Updating a Department Record 92
Figure 71: Updated Department Record shown in table 93
Figure 72: Department Table before deleting a Department Record..... 94
Figure 73: Deleting a Department Record 94
Figure 74: Department Table after deleting a Department Record 95
Figure 75: Reading data from Teacher Table 96
Figure 76: Teacher Table 97
Figure 77: Inserting New Teacher Record 97
Figure 78: Inserting New Teacher Record shown in table 98
Figure 79: Teacher Table 99
Figure 80: Updating a Teacher Record 99
Figure 81: Updated Teacher Record shown in table 100
Figure 82: Teacher Table before deleting a Teacher Record 101
Figure 83: Deleting a Teacher Record 101
Figure 84: Teacher Table after deleting a Teacher Record 102
Figure 85: Reading data from Address Table 103
Figure 86: Address Table 104
Figure 87: Inserting New Address Record 104
Figure 88: Inserting New Address Record shown in table 105
Figure 89: Address Table 106
Figure 90: Updating an Address Record..... 106
Figure 91: Updated Address Record shown in table 107
Figure 92: Testing of update operation in Address Table 107
Figure 93: Address Table before deleting an Address Record 108
Figure 94: Deleting an Address Record 108
Figure 95: Address Table after deleting an Address Record 109
Figure 96: Reading data from Module Table 110
Figure 97: Module Table 111
Figure 98: Inserting New Module Record 111
Figure 99: Inserting New Module Record shown in table 112
Figure 100: Module Table 113

Figure 101: Updating a Module Record 113
Figure 102: Updated Module Record shown in table 114
Figure 103 : Module Table before deleting a Module Record 115
Figure 104: Deleting a Module Record 115
Figure 105: Module Table after deleting a Module Record 116
Figure 106: Failed cases 117
Figure 107: User Manual of Student Simple Form 118
Figure 108: User Manual of Department Simple Form 120
Figure 109: User Manual of Teacher Simple Form 122
Figure 110: User Manual of Address Simple Form 124
Figure 111: User Manual of Module Simple Form 126
Figure 112: User Manual of Teacher Module Mapping Form 128
Figure 113: User Manual of Student Fee Payment Form 130
Figure 114: User Manual of Student Assignment Form 132

Table of Tables

Table 1: Data dictionary of Address Table 20
Table 2: Data dictionary of Assignment Table 21
Table 3: Data dictionary of Departments Table 22
Table 4: Data dictionary of Person_Address_info Table 22
Table 5: Data dictionary of Module Table 23
Table 6: Data dictionary of Person Table 24
Table 7: Data dictionary of Result_info Table 25
Table 8: Data dictionary of Student Table 26
Table 9: Data dictionary of Student_Attendance_info Table 27
Table 10: Data dictionary of Student_Fee_Payment Table 28
Table 11: Data dictionary of Module_Student_info Table 29
Table 12: Data dictionary of Teacher Table 30
Table 13: Data dictionary of Teacher_Module_info Table 31
Table 14: Testing of create operation in Student Table 85
Table 15: Testing of update operation in Student Table 87
Table 16: Testing of delete operation in Student Table 89
Table 17: Testing of create operation in Department Table 91
Table 18: Testing of update operation in Department Table 93
Table 19 : Testing of delete operation in Department Table 95
Table 20: Testing of create operation in Teacher Table 98
Table 21: Testing of update operation in Teacher Table 100
Table 22: Testing of delete operation in Teacher Table 102
Table 23: Testing of create operation in Address Table 105
Table 24: Testing of delete operation in Address Table 109

Table 25: Testing of create operation in Module Table 112
Table 26: Testing of update operation in Module Table..... 114
Table 27: Testing of delete operation in Module Table 116

Introduction

As the world has been completely digitizing daily at a rapid pace, people do not realize about the significant affect in our daily life. Even though people do not know much about database, they use it extensively such as in grocery store, bank, restaurant, hospital, and many more to store and keep track of customer data. From a customer point of view, most of the people are unaware about the existence of a database while utilizing an application. Database allows a user to store data in a simple manner and retrieve the information in need. It has the capability of storing huge amount of data at once and enables the user to modify the data inside the database.

According to the coursework paper provided to us, we are asked to analyze, design, and implement a web-based database application based on the given business case study. As per the given scenario, a database should be created for the Berkeley college to keep track of the student details, module details, teacher details and many more. The Berkeley college has several internal departments as well as modules and their specific staffs. For the development of the entire web application, I have used SQL developer for creating the database, data modeler for designing the E-R diagram and ASP.NET for the coding part.

Oracle SQL developer can be defined as an open-source graphical tool that enables the user to enhance their productivity as it simplifies the database development tasks. By the help of SQL developer, a user can browse database objects, run SQL statements and SQL scripts, as well as edit and debug PL and SQL statements. Formerly known as Project Raptor, it is a JAVA-based and works in Windows, Unix, and Linux. (FYICenter, 2011).

Oracle SQL data modeler is another interesting graphical tool that enables the user to create, browse and edit logical, relational, multi- dimensional and many other types of data models. It can be used in both traditional and Cloud environments. Oracle SQL data modeler also provides forward and reverse engineering capabilities and supports collaborative development through integrated source code control. (Oracle, 2022).

Visual Studio is an Integrated Development Environment (IDE) that is used to develop GUI console, web applications, mobile applications and many more. It is not language specific and enables the user to write code in C sharp, C++, Visual Basic, Python and many other languages.

In total, Visual Studio can support 36 different programming languages and available in both Windows and MacOS. Visual Studio is the best suitable IDE for working with Oracle database and C sharp programming language. Due to its wide range of features, user can easily create simple and complex forms by using its drag and drop feature. (GeeksforGeeks, 2019)

Textual Analysis

Analyzing the given case study in the coursework, I have created textual analysis for demonstrating the relationship between every two entities. Along with that, a short description about the cardinality between those specific entities has also been mentioned below the figures.

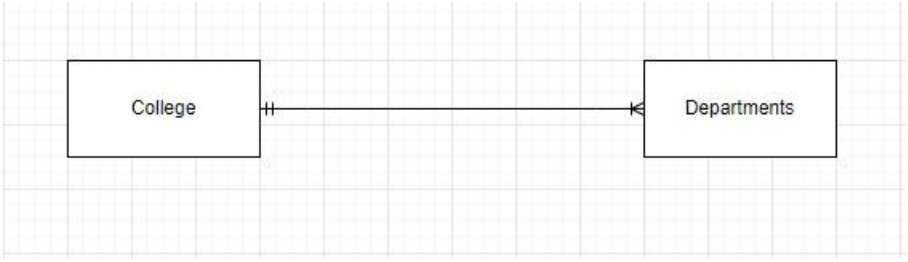


Figure 1: Textual Analysis of College and Departments

According to the up-given diagram, it demonstrates a one-to-many relation between college entity and departments entity. A college can have control over multiple departments.

Figure 2: Textual Analysis of Department and Examinations

According to the up-given diagram, it demonstrates a one-to-many relation between Department entity and examinations entity. A single department will conduct and manage multiple student examinations.

Figure 3: Textual Analysis of Department and Assignment

According to the up-given diagram, it demonstrates a one-to-many relation between Department entity and Assignment entity. A single department will assign and manage multiple assignments for the college students.

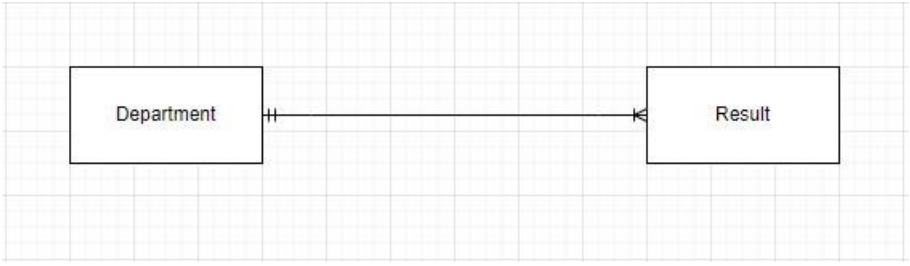


Figure 4: Textual Analysis of Department and Result

According to the up-given diagram, it demonstrates a one-to-many relation between Department entity and Result entity. A single department will conduct and manage results of multiple college students.

Figure 5: Textual Analysis of Department and Fee

According to the up-given diagram, it demonstrates a one-to-many relation between Department entity and Fee entity. A single department will manage the student fees record of multiple college students.

Figure 6: Textual Analysis of Department and Attendance

According to the up-given diagram, it demonstrates a one-to-many relation between Department entity and Attendance entity. The relation describes that a single department will look over the attendance record of multiple college students to decide whether a single college student is eligible to take module examination or assignment or not.

Figure 7: Textual Analysis of Student and Module

According to the up-given diagram, it demonstrates a many-to-many relation between Student entity and Module entity. As per the relation, Multiple students can read a single module whereas a single student can study multiple modules at a time.

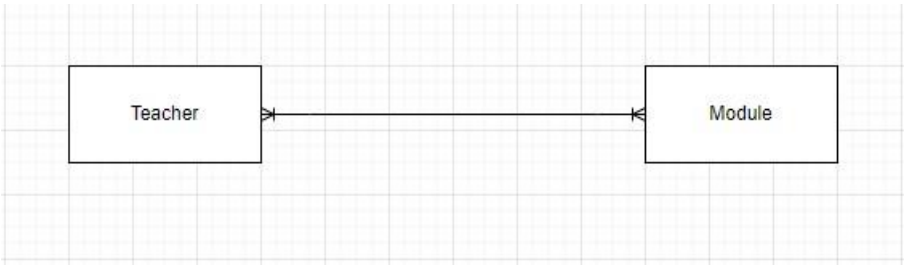


Figure 8: Textual Analysis of Teacher and Module

According to the up-given diagram, it demonstrates a many-to-many relation between Teacher entity and Module entity. As per the relation, a single teacher can be allocated to multiple modules whereas a single module can be taught by multiple teachers.

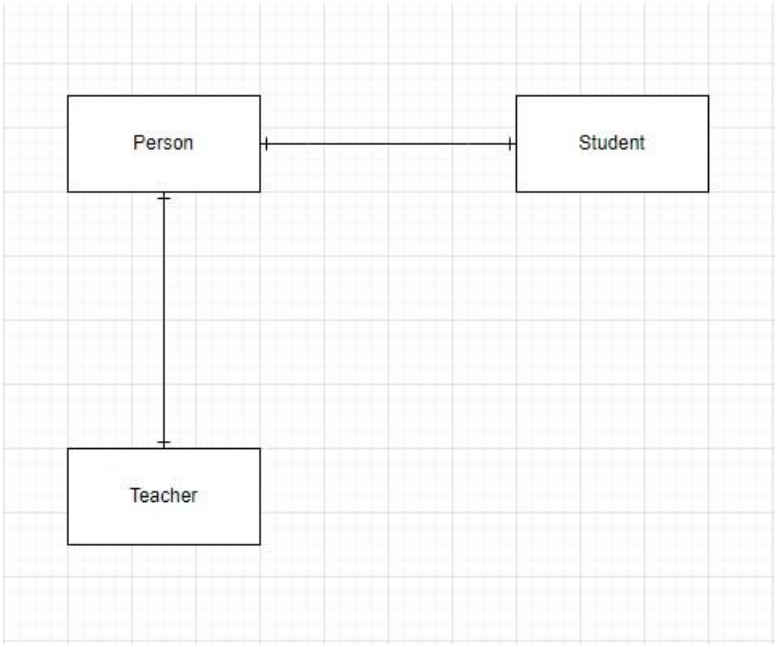


Figure 9: Textual Analysis of Person, Student and Techer

For the last case of the scenario, a person can be a student and be a teacher after graduating college. To satisfy this condition, I proposed a separate entity named Person. Through it, a person can have two separate ids' i.e., Teacher id and Student id which will preserve the individual's data in their relevant tables. However, a person cannot act as both at once as per the condition.

ERD from Case Study

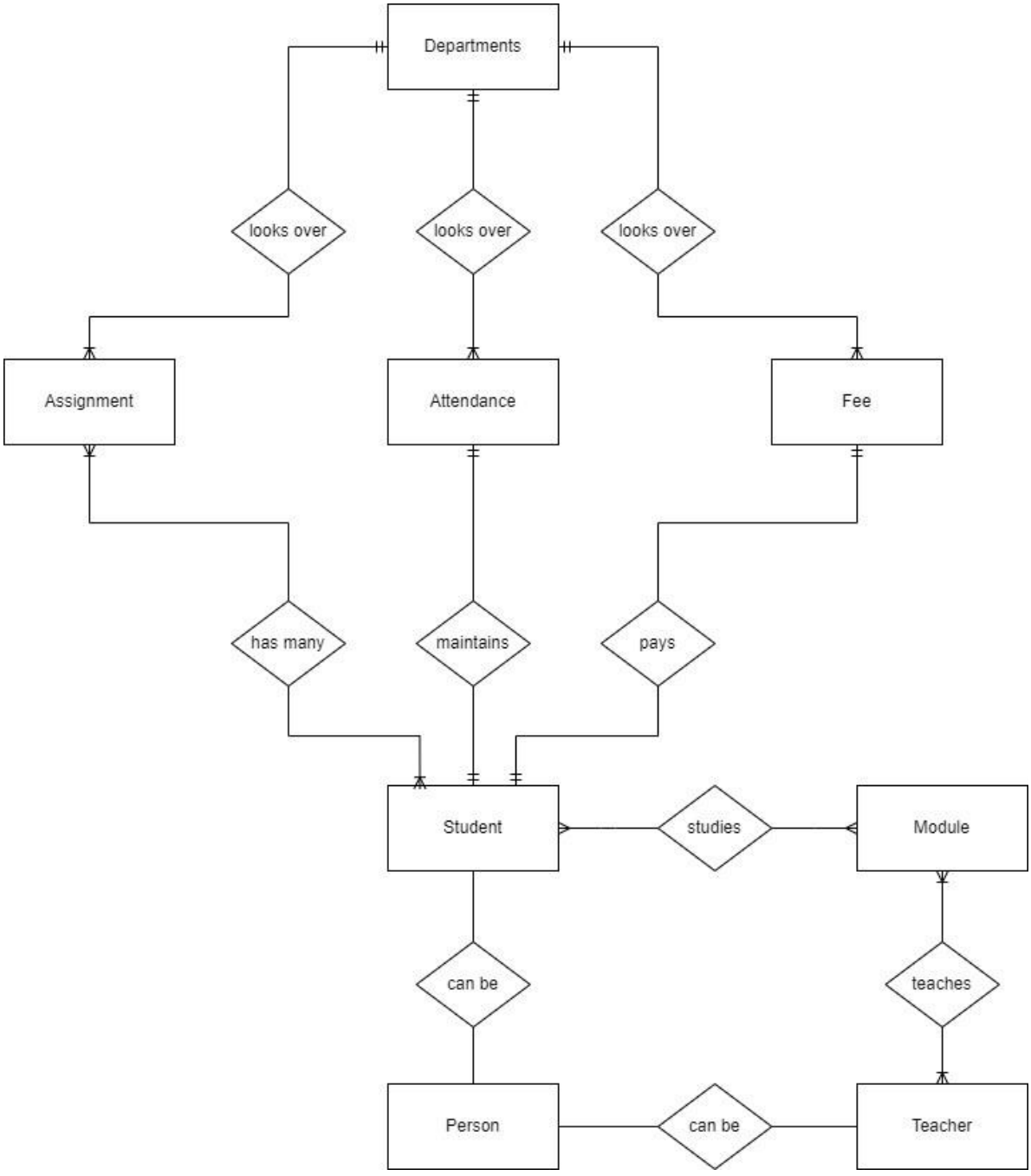


Figure 10: Initial Raw ERD from Case Study

Normalization

Normalization can be referred as a technique that is used to reduce data redundancy and eliminate undesirable characteristics such as Insertion, Update and Deletion anomalies. In simple words, Normalization is implemented in the database to eliminate redundant data and ensure data has been logically stored inside the database. By following the rules of normalization, larger tables inside a database are divided into further smaller tables and linked by using relationships. Inside the normalization process, there are a total of seven different normal forms. However, we have only leaned till the third normal form as in most of the practical applications, normalization achieves its best in the third normal form (Peterson, 2021). The entire extended theory of normalization till the third normal form is described briefly below:

UNF

Un-normalized form is the simplest database model that suffer problems like data redundancy. In this form, repeating groups and repeating data are identified and kept in separated by using curly braces. Even though it lacks the efficiency of database normalization, it can deal with the complex data structures and restricting the data is much easier. (GeeksforGeeks, 2020)

1NF

The data model is set to be in the First Normal Form if only the relational table does not consist of multivalued attributes but only hold atomic or single-valued attributes. In the First normal form, each of the column name should be unique and every table must contain a primary key. (GeeksforGeeks, 2020)

2NF

Initially the data model must be in its First Normal Form to obtain its Second Normal form. Every single partial dependency must be removed from the data model to be in Second Normal Form (GeeksforGeeks, 2020). Partial Dependency always occurs when a non-prime attribute is functionally dependent on part of a candidate key.

3NF

For the data model to be in its Third Normal form, it should initially be in its Second Normal Form. After that, transitive dependencies inside the data model are identified and removed. If the data model does not contain any type of Transitive dependency, then it will automatically be in its Third Normal Form (GeeksforGeeks, 2020). Transitive Dependency occurs when an indirect relationship is formed between data elements where a non-prime attribute is functionally dependent on another non-prime attribute.

Normalization of Figure 1

Fig 1. Example of Teacher allocation list

S.N.	Teacher Name	Address	Email	Module Code	Module Name	Credit Hours
1	Saul Goodman	595 Green Lake Road Black Lake 9115 Lake Street Harrietsfield	Saulthegoodman@abc.edu.np	CC12	Data Structure and Algorithm	30
2	Walter White	696 Madison St. Pierrefonds	whitywalker@abc.edu.np	CC12	Data Structure and Algorithm	30
3	Santana Lopez	6 Valley View Street Griffintown	Santanalopez@abc.edu.np	CC49	Engineering Thermodynamic	60
4	Rust Cohle	89 Coffee Dr. Plaster Rock	rustycohle@abc.edu.np	SG101 TG405	Software engineer Data Analysis	30 50

Figure 11: Figure 1 for normalization

Un-normalized Form (UNF)

For the starting phase of normalization, I had to take the up given figure into its unnormalized form. I separated the repeating groups and repeating data from the table and kept it inside the same entity i.e., Teacher. In my case, the repeating data are Address, Module Code and Module Name and Credit Hours. After classifying the repeating data, it is separated and kept in curly braces. For the repeating groups, S.N is the unique identifier or the primary key. For the better visualization of the primary key, it is kept in bold and underline format below.

Teacher – (**S.N**, Teacher_name, E-mail, {Address_id, Address}, {Module_code, Module_name, Credit_hours})

First Normal Form (1NF)

For the second phase of the normalization, Un-normalized form of the table figure is taken into its First Normal Form. Inside the First Normal Form, repeating group and repeating data are separated into different entities. Talking about the repeating group, all the attributes are kept under a single entity i.e., Teacher. Apart from that, Address and Module entities are also created which includes the attributes of repeating data. The tables created in the First Normal form of the table figure are listed below:

Entities:

Teacher-1 (**S.N**, Teacher_name, E-mail)

Address-1 (**Address_id**, Address, **S.N***)

Module-1 (**Module_code**, Module_name, Credit_hours, **S.N***)

Second Normal Form (2NF)

After taking the table figure into its First Normal Form, we should take it to the Second Normal form for clearing partial dependencies in its entities. For checking the partial dependencies inside a table, it must consist of a composite primary key. Other tables which consist of only a single unique identifier are considered to already be in its Second normal form. In my case, Address and Module entities contain multiple numbers of unique identifiers. Both tables are checked for any type of partial dependencies in the following way:

For Address:

Address_id → Address

Address_id, **S.N*** →

For Module:

Module_code → Module_name, Credit_hours

Module_code, **S.N*** →

After checking the partial dependencies in both tables, new entity i.e., Teacher_info was created. Now, the figure is in its Second Normal form (2NF).

Entities:

Teacher-2 (**S.N**, Teacher_name, E-mail)

Address-2 (**Address_id**, Address)

Module-2 (**Module_code**, Module_name, Credit_hours)

Teacher_info-2 (**S.N*, Address_id*, Module_code***)

Third Normal Form (3NF)

In the last phase of the normalization process, transitive dependencies are removed from the entities. For the entity to be eligible for checking any kind of transitive dependencies, it must contain more than one non-key attribute. In my case, Teacher and Module table consists of more than one non-key attributes. Other tables which contain only a single non-key attribute are automatically considered to be in its third normal form. Both tables are checked for any type of transitive dependencies in the following way:

For Teacher:

S.N → Teacher_name → **X**

S.N → E-mail → **X For**

Module:

Module_code → Module_name → **X**

Module_code → Credit_hours → **X**

After checking for transitive dependencies in both tables, there were no transitive dependencies inside the tables and remained same as it was in its second normal form.

With the completion of the third normal form, the normalization process is completed, and figure table does not contain any kind of functional dependencies or data redundancy. The final tables of the figure after normalization are listed below:

Final Entities:

Teacher-3 (**S.N**, Teacher_name, E-mail)

Address-3 (**Address_id**, Address)

Module-3 (**Module_code**, Module_name, Credit_hours) **Teacher_info-3**

(**S.N***, **Address_id***, **Module_code***)

Normalization of Figure 2

Fig 2. Example of Assignment and Examination Results

Student ID: 149893

Student Name: Mr. William Ishee

Student Address: 2508 Shinn Street New York

Module Code	*Module Name*	*Assignment Type*	*Grade*	*Status*
CC12	Data Structure and Algorithm	*Coursework*	*A*	*Pass*
CC49	Engineering Thermodynamic	*Coursework*	*B*	*Pass*
CC49	Engineering Thermodynamic	*Written Exam*	*F*	*Fail*
SG101	Software engineer	*Individual Assignment*	*B+*	*Pass*
SG101	Software engineer	*Group Assignment*	*B*	*Pass*
SG101	Software engineer	*Unseen Examination*	*A*	*Pass*

Figure 12: Figure 2 for normalization

Un-normalized Form (UNF)

To start the normalization process of the given figure, I took the figure in its unnormalized form by combining all the attributes inside a single entity, Student. Inside the table, I separated the repeating group and repeating data and kept the repeating data inside curly braces. For the repeating groups, Student_ID is the unique identifier or the primary key. I have added an additional attribute, Assignment_ID in the repeating data which will act as a unique identifier. For the better visualization of the primary key, it is kept in bold and underline format below.

Student – (**Student_ID**, Student_Name, Student_Address, {Module_Code, Module_Name}, {Assignment_ID, Assignment_Type, Grade, Status})

First Normal Form (1NF)

After taking the table into its Un-normalized form, it reaches in the second phase of normalization. In this phase, repeating data and repeating group are separated from a singular entity and kept in separate entities. The attributes of the repeating group are kept inside an entity named Student whereas the repeating data are kept on other tables that are Module and Assignment. The tables created in the First Normal form of the table figure are listed below:

Entities:

Student-1 (**Student_ID**, Student_Name, Student_Address)

Module-1 (**Module_Code**, Module_Name, **Student_ID***)

Assignment-1 (**Assignment_ID**, Assignment_Type, Grade, Status, **Student_ID***)

Second Normal Form (2NF)

As the table reaches it First Normal form, the attributes are separated into different tables from a singular entity. In the second normal form, Partial dependencies are checked and removed from the tables. For checking the partial dependencies inside a table, it must consist of a composite primary key. Other tables which consist of only a single unique identifier are considered to already be in its Second normal form. In my case, Module and Assignment entities contain multiple numbers of unique identifiers. Both tables are checked for any type of partial dependencies in the following way:

For Module:

Module_Code → Module_Name **Module_Code,**

Student_ID* → **For Assignment:**

Assignment_ID → Assignment_Type

Assignment_ID, Student_ID* → Grade, Status

After checking the partial dependencies in both tables, new entities i.e., Teacher_info was created. Now, the figure is in its Second Normal form (2NF).

Entities:

Student-2 (**Student_ID**, Student_Name, Student_Address)

Module-2 (**Module_Code**, Module_Name)

Student_Module_info-2 (**Module_Code***, **Student_ID***)

Assignment-2 (**Assignment_ID**, Assignment_Type)

Student_Assignment_info-2 (**Assignment_ID***, **Student_ID*,** Grade, Status)

Third Normal Form (3NF)

After the completion of the second normal form, the figure reaches its last phase of the normalization process, third normal form. In this phase, Transitive dependencies are removed from the tables. For the entity to be eligible for checking any kind of transitive dependencies, it must contain more than one non-key attribute. In my case, Student and Student_Assignment_info table consists of more than one non-key attributes. Other tables which contain only a single non-key attribute are automatically considered to be in its third normal form.

For Teacher:

Student_ID* → Student_Name → **X**

Student_ID* → Student_Address → **X**

For Student_Assignment_info:

Assignment_ID*, Student_ID* → Grade → Status

After checking for transitive dependencies in both tables, Student_Assignment_info consisted of a transitive dependency. Hence, the attribute was separated into two different tables creating a new entity, Assignment_Status.

With the completion of the third normal form, the normalization process is completed, and figure table does not contain any kind of functional dependencies or data redundancy. The final tables of the figure after normalization are listed below:

Final Entities:

Student-3 (**Student_ID**, Student_Name, Student_Address)

Module-3 (**Module_Code**, Module_Name)

Student_Module_info-3 (**Module_Code***, **Student_ID***)

Assignment-3 (**Assignment_ID**, Assignment_Type)

Assignment_Status-3 (**Status_ID,** Status)

Student_Assignment_info-3 (**Assignment_ID***, **Student_ID*,** Grade, **Status_ID***)

Integration and Assumption

According to the case study, I have prepared a list of assumptions based on it that are briefly listed below:

- The college will have the privilege to look over multiple departments.
- Among the several departments, each department will look over a specific program of the student such as student examination, fee, and attendance.
- A person inside the college can either be a student or a teacher but cannot be both at a single time.
- A person can have multiple addresses whereas multiple persons can have the same specific address.
- A student can enroll into multiple modules at a single time.
- A single module can be studied by multiple students at a single time.
- A teacher can be assigned for teaching multiple modules at a single time.
- Multiple teachers can teach the same module at the same time.
- A student is only able to take any kind of module assignment or examination if he/she has already paid fees and must have attendance count over 80%.
- A student can also be a teacher in the same college after his/her graduation.

After the normalization process, I have combined the final entities from Case Study, Figure 1 and Figure 2 and integrated them into a single ER diagram. Apart from that, I have also added some new entities which are not presented in the up given normalization process such as Exam_info, Student_Attendance_info, Student_Fee_Payment and many more, to satisfy the condition given in the case scenario. The final entities for developing my database are briefly listed below along with primary and foreign keys.

Notation:

Bold and Underlined "_" = Primary Key

Bold and Asterisk sign "*" = Foreign Key

Final Entities:

Departments (Department_ID, Department_Name**)**

Person (Person_ID, First_Name, Last_Name, E-mail**)**

Address (Address_ID, Country, City, Postal_Code**)**

Person_Address_info (Person_ID*, Address_ID*)

Student (Person_ID*, Student_Year**)**

Student_Fee_Payment (Payment_ID, Amount, Year, Payment_Date**, Student_ID*, Department_ID*)**

Teacher (Person_ID*, Teacher_Type, Salary**)**

Module (Module_ID, Module_Code, Module_Name, Module_Teaching_Days, Module_Credit**)**

Assignment (Assignment_ID, Assignment_Type, **Department_ID***, **Module_ID*)**

Module_Student (Module_ID*, Student_ID*,)

Teacher_Module_info (Teacher_ID*, Module_ID*)

Student_Attendance_info (Student_ID*, Module_ID*, Attendance_Date, **Department_ID*)**

Result_info (Result_ID, Grade, Status**, Student_ID*, Assignment_ID*)**

Final ERD

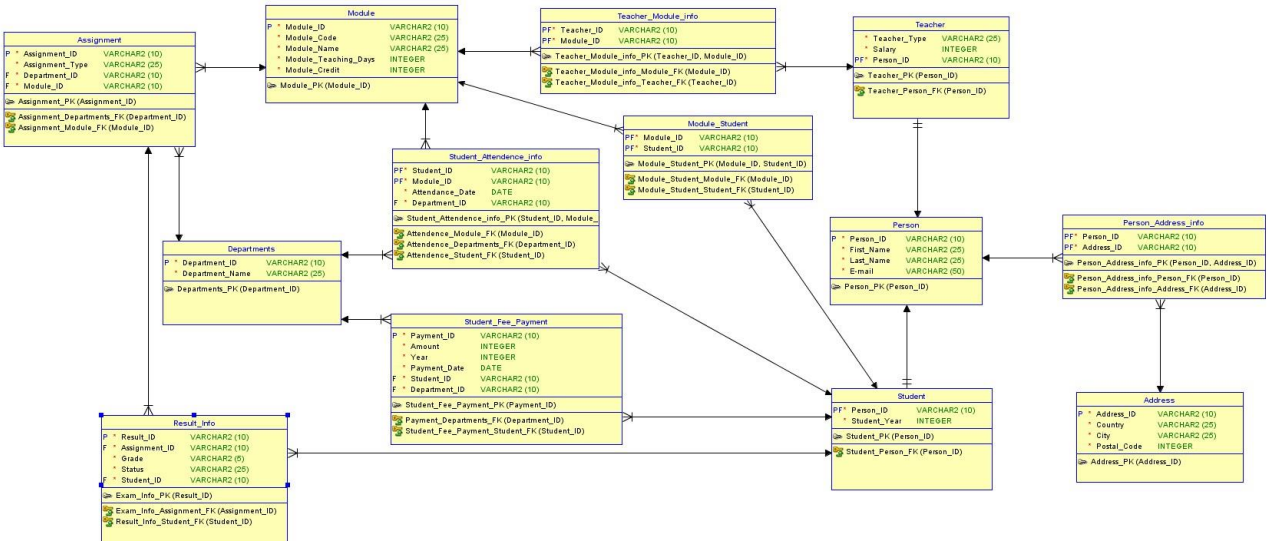


Figure 13: Final ER diagram of the database

Data Dictionary

After finalizing the tables for my database, I have created a data dictionary for every single entity for better understanding of attributes inside the table along with their data types, size, unique constraints, description, and example values.

Address Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Address_ID	VARCHAR2	10	Primary Key (PK)			To uniquely identify each person's address.	Add1
Country	VARCHAR2	25	Not Null			To store the person's country name	Nepal
City	VARCHAR2	25	Not Null			To store the person's city name	Dharan
Postal_Code	NUMBER	38	Not Null			To store the person's postal code	44600

Table 1: Data dictionary of Address Table

Assignment Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Assignment_ID	VARCHAR2	10	Primary Key (PK)			To uniquely identify each assignment	Ass1
Assignment_Type	VARCHAR2	25	Not Null			To store the assignment's type.	MCQ
Department_ID	VARCHAR2	10	Foreign Key (FK)	Department	Department_ID	To store the assignment's allocated department	D1
Module_ID	VARCHAR2	10	Foreign Key (FK)	Module	Module_ID	To store the assignment's related module	M1

Table 2: Data dictionary of Assignment Table

Departments Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Department_ID	VARCHAR2	10	Primary Key (PK)			To uniquely identify each department	D1
Department_Name	VARCHAR2	25	Not Null			To store the department's name	Human Resources

Table 3: Data dictionary of Departments Table

Person_Address_info Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Person_ID	VARCHAR2	10	Primary Key (PK), Foreign Key (FK)	Person	Person_ID	To uniquely identify each person	P1
Address_ID	VARCHAR2	10	Primary Key (PK), Foreign Key (FK)	Address	Address_ID	To uniquely identify each person's address	Add1

Table 4: Data dictionary of Person_Address_info Table

Module Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Module_ID	VARCHAR2	10	Primary Key (PK)			To uniquely identify each module.	M1
Module_Code	VARCHAR2	25	Not Null			To store the module's specific code	CC6051NI
Module_Name	VARCHAR2	25	Not Null			To store the module's name	Database
Module_Teaching_Days	NUMBER	38	Not Null			To store the module's teaching days	8
Module_Credit	NUMBER	38	Not Null			To store the module's credit	50

Table 5: Data dictionary of Module Table

Person Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Person_ID	VARCHAR2	10	Primary Key (PK)			To uniquely identify each person.	P1
First_Name	VARCHAR2	25	Not Null			To store the person's first name.	Ram
Last_Name	VARCHAR2	25	Not Null			To store the person's last name.	Limbu
E-mail	VARCHAR2	50	Not Null			To store the specific person's email.	abc@gmail.com

Table 6: Data dictionary of Person Table

Result_info Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Result_ID	VARCHAR2	10	Primary Key (PK)			To uniquely identify each person's result.	R1
Grade	VARCHAR2	5	Not Null			To store the person's specific grade	A
Status	VARCHAR2	25	Not Null			To store the specific person's grade status	Good
Assignment_ID	VARCHAR2	10	Foreign Key (FK)	Assignment	Assignment _ID	To uniquely identify a specific assignment	Ass1
Student_ID	VARCHAR2	10	Foreign Key (FK)	Student	Person _ID	To uniquely identify each person who is a student	P1

Table 7: Data dictionary of Result_info Table

Student Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Person_ID	VARCHAR2	10	Primary Key (PK), Foreign Key (FK)	Person	Person_ID	To uniquely identify each person.	P1
Student_Year	NUMBER	38	Not Null			To store the student's specific year	1

Table 8: Data dictionary of Student Table

Student_Attendance_info Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Student_ID	VARCHAR 2	10	Primary Key (PK), Foreign Key (FK)	Student	Person_ID	To uniquely identify each person who is a student	P1
Module_ID	VARCHAR 2	10	Primary Key (PK), Foreign Key (FK)	Module	Module_ID	To store the student's specific module	M1
Attendance_ Date	Date	38	Not null			To store the attendance date of a specific student	2/12/2003
Department_ID	VARCHAR 2	10	Foreign Key (FK)	Department	Department _ID	To store the assignment's allocated department	D1

Table 9: Data dictionary of Student_Attendance_info Table

Student_Fee_Payment Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Payment_ID	VARCHAR 2	10	Primary Key (PK)			To uniquely identify each student's fee payment	Pay1
Amount	NUMBER	38	Not Null			To store the student's payment amount	1,00,000
Year	NUMBER	38	Not null			To store the specific year for student's payment	2
Payment_Date	DATE		Not null			To store the specific date of student's fee payment	02/13/1999
Student_ID	VARCHAR 2	10	Foreign Key (FK)	Student	Person_ID	To uniquely identify each person who is a student	P1

Department_ID	VARCHAR 2	10	Foreign Key (FK)	Department	Department_ ID	To store the assignment's allocated department	D1

Table 10: Data dictionary of Student_Fee_Payment Table

Module_Student Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Module_ID	VARCHAR 2	10	Primary Key (PK), Foreign Key (FK)	Module	Module_ID	To uniquely identify specific teacher's module	M1
Student_ID	VARCHAR 2	10	Primary Key (PK), Foreign Key (FK)	Student	Person_ID	To uniquely identify each person who is a student	P1

Table 11: Data dictionary of Module_Student_info Table

Teacher Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Person_ID	VARCHAR2	10	Primary Key (PK), Foreign Key (FK)	Person	Person_ID	To uniquely identify each person.	P1
Teacher-Type	VARCHAR2	25	Not Null			To store the specific teacher's role or type	Tutor
Salary	NUMBER	38	Not Null			To store the specific teacher's salary	30,000

Table 12: Data dictionary of Teacher Table

Teacher_Module_info Table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Values
Teacher_ID	VARCHAR2	10	Primary Key (PK), Foreign Key (FK)	Teacher	Person_ID	To uniquely identify each person who is a teacher	P1
Module_ID	VARCHAR2	10	Primary Key (PK), Foreign Key (FK)	Module	Module_ID	To uniquely identify specific teacher's module	M1

Table 13: Data dictionary of Teacher_Module_info Table

Script

Generation of Database

```
C:\Users\User>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.2.0 Production on Sat Mar 5 12:53:10 2022

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> CREATE USER Berkeley_College identified by coursework;

User created.

SQL>
```

Figure 14: Creating a new user in oracle, Berkeley_College

```
C:\Users\User>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.2.0 Production on Sat Mar 5 12:53:10 2022

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> CREATE USER Berkeley_College identified by coursework;

User created.

SQL> GRANT ALL PRIVILEGES TO Berkeley_College;

Grant succeeded.

SQL>
```

Figure 15: Granting all the privileges to Berkeley_College user

Create Statements

Create Statement for Address table

```
CREATE TABLE address ( address_id

VARCHAR2(10) NOT NULL, country

VARCHAR2(25) NOT NULL, city

VARCHAR2(25) NOT NULL,

postal_code INTEGER NOT NULL

);

ALTER TABLE address ADD CONSTRAINT address_pk PRIMARY KEY ( address_id );
```

Create Statement for Assignment table

```
CREATE TABLE assignment ( assignment_id

VARCHAR2(10) NOT NULL, assignment_type

VARCHAR2(25) NOT NULL, department_id

VARCHAR2(10) NOT NULL, module_id

VARCHAR2(10) NOT NULL

);

ALTER TABLE assignment ADD CONSTRAINT assignment_pk PRIMARY KEY ( assignment_id );
```

Create Statement for Departments table

```
CREATE TABLE departments ( department_id
VARCHAR2(10) NOT NULL,
department_name VARCHAR2(25) NOT
NULL

);
ALTER TABLE departments ADD CONSTRAINT departments_pk PRIMARY KEY ( department_id );
```

Create Statement for Module table

```
CREATE TABLE module ( module_id
VARCHAR2(10) NOT NULL, module_code
VARCHAR2(25) NOT NULL, module_name
VARCHAR2(25) NOT NULL,
module_teaching_days INTEGER NOT NULL,
module_credit INTEGER NOT NULL
);
ALTER TABLE module ADD CONSTRAINT module_pk PRIMARY KEY ( module_id );
```

Create Statement for Person table

```
CREATE TABLE person ( person_id

VARCHAR2(10) NOT NULL, first_name

VARCHAR2(25) NOT NULL, last_name

VARCHAR2(25) NOT NULL,

"E-mail"   VARCHAR2(50) NOT NULL

);

ALTER TABLE person ADD CONSTRAINT person_pk PRIMARY KEY ( person_id );
```

Create Statement for Person_Address_info table

```
CREATE TABLE person_address_info ( person_id

VARCHAR2(10) NOT NULL, address_id

VARCHAR2(10) NOT NULL

);

ALTER TABLE person_address_info ADD CONSTRAINT person_address_info_pk PRIMARY KEY
(person_id, address_id );
```

Create Statement for Result_info table

```
CREATE TABLE result_info ( result_id

VARCHAR2(10) NOT NULL, assignment_id

VARCHAR2(10) NOT NULL, grade

VARCHAR2(5) NOT NULL, status
```

```
VARCHAR2(25) NOT NULL, student_id

VARCHAR2(10) NOT NULL

);

ALTER TABLE result_info ADD CONSTRAINT exam_info_pk PRIMARY KEY ( result_id );
```

Create Statement for Student table

```
CREATE TABLE student ( person_id

VARCHAR2(10) NOT NULL,

student_year INTEGER NOT NULL

);

ALTER TABLE student ADD CONSTRAINT student_pk PRIMARY KEY ( person_id );
```

Create Statement for Student_Attendance_info table

```
CREATE TABLE student_attendence_info (

student_id      VARCHAR2(10) NOT NULL,

module_id       VARCHAR2(10) NOT NULL,

attendance_date DATE NOT NULL, department_id

VARCHAR2(10) NOT NULL

);

ALTER TABLE student_attendence_info ADD CONSTRAINT student_attendence_info_pk PRIMARY KEY
(student_id, module_id );
```

Create Statement for Student_Fee_Payment table

```
CREATE TABLE student_fee_payment (
payment_id VARCHAR2(10) NOT NULL,
amount INTEGER NOT NULL, year
INTEGER NOT NULL, payment_date
DATE NOT NULL, student_id
VARCHAR2(10) NOT NULL,
department_id VARCHAR2(10) NOT NULL
);
ALTER TABLE student_fee_payment ADD CONSTRAINT student_fee_payment_pk PRIMARY KEY (
payment_id );
```

Create Statement for Teacher table

```
CREATE TABLE teacher ( teacher_type
VARCHAR2(25) NOT NULL, salary
INTEGER NOT NULL, person_id
VARCHAR2(10) NOT NULL
);
ALTER TABLE teacher ADD CONSTRAINT teacher_pk PRIMARY KEY ( person_id );
```

Create Statement for Teacher_Module_info table

```
CREATE TABLE teacher_module_info ( teacher_id

VARCHAR2(10) NOT NULL, module_id

VARCHAR2(10) NOT NULL

);

ALTER TABLE teacher_module_info ADD CONSTRAINT teacher_module_info_pk PRIMARY KEY
(teacher_id, module_id );
```

Create Statement for Module_Student table

```
CREATE TABLE module_student ( module_id

VARCHAR2(10) NOT NULL, student_id

VARCHAR2(10) NOT NULL

);

ALTER TABLE module_student ADD CONSTRAINT module_student_pk PRIMARY KEY ( module_id,
student_id );
```

Alter Statements of all tables for adding foreign keys

```
ALTER TABLE assignment

ADD CONSTRAINT assignment_departments_fk FOREIGN KEY ( department_id ) REFERENCES

departments ( department_id );


ALTER TABLE assignment

ADD CONSTRAINT assignment_module_fk FOREIGN KEY ( module_id )
```

```
REFERENCES module ( module_id );


ALTER TABLE student_attendence_info

ADD CONSTRAINT attendence_departments_fk FOREIGN KEY ( department_id ) REFERENCES

departments ( department_id );


ALTER TABLE student_attendence_info

ADD CONSTRAINT attendence_module_fk FOREIGN KEY ( module_id )

REFERENCES module ( module_id );


ALTER TABLE student_attendence_info

ADD CONSTRAINT attendence_student_fk FOREIGN KEY ( student_id )
REFERENCES student ( person_id );


ALTER TABLE result_info

ADD CONSTRAINT exam_info_assignment_fk FOREIGN KEY ( assignment_id )

REFERENCES assignment ( assignment_id );


ALTER TABLE module_student

ADD CONSTRAINT module_student_module_fk FOREIGN KEY ( module_id )

REFERENCES module ( module_id );
```

```
ALTER TABLE module_student
ADD CONSTRAINT module_student_student_fk FOREIGN KEY ( student_id )
REFERENCES student ( person_id );

ALTER TABLE student_fee_payment
ADD CONSTRAINT payment_departments_fk FOREIGN KEY ( department_id )
REFERENCES departments ( department_id );

ALTER TABLE person_address_info
ADD CONSTRAINT person_address_info_address_fk FOREIGN KEY ( address_id )
REFERENCES address ( address_id );

ALTER TABLE person_address_info
ADD CONSTRAINT person_address_info_person_fk FOREIGN KEY ( person_id )
REFERENCES person ( person_id );

ALTER TABLE result_info
ADD CONSTRAINT result_info_student_fk FOREIGN KEY ( student_id )
REFERENCES student ( person_id );

ALTER TABLE student_fee_payment
ADD CONSTRAINT student_fee_payment_student_fk FOREIGN KEY ( student_id )
REFERENCES student ( person_id );
```

```
ALTER TABLE student

ADD CONSTRAINT student_person_fk FOREIGN KEY ( person_id )

REFERENCES person ( person_id );


ALTER TABLE teacher_module_info

ADD CONSTRAINT teacher_module_info_module_fk FOREIGN KEY ( module_id )

REFERENCES module ( module_id );


ALTER TABLE teacher_module_info
ADD CONSTRAINT teacher_module_info_teacher_fk FOREIGN KEY ( teacher_id ) REFERENCES

teacher ( person_id );


ALTER TABLE teacher

ADD CONSTRAINT teacher_person_fk FOREIGN KEY ( person_id )

REFERENCES person ( person_id );
```

After creating the Final ER-diagram of the database in the SQL data modeler application, I generated a DDL script of the ER- diagram. Basically, the DDL script is a subset of SQL that Is used for describing data and its relationship in a database. After the generating the DDL script, I executed the script in Oracle SQL developer which develops a database based on the data provided by the script. I have provided the screenshots of the script execution in SQL developer below.

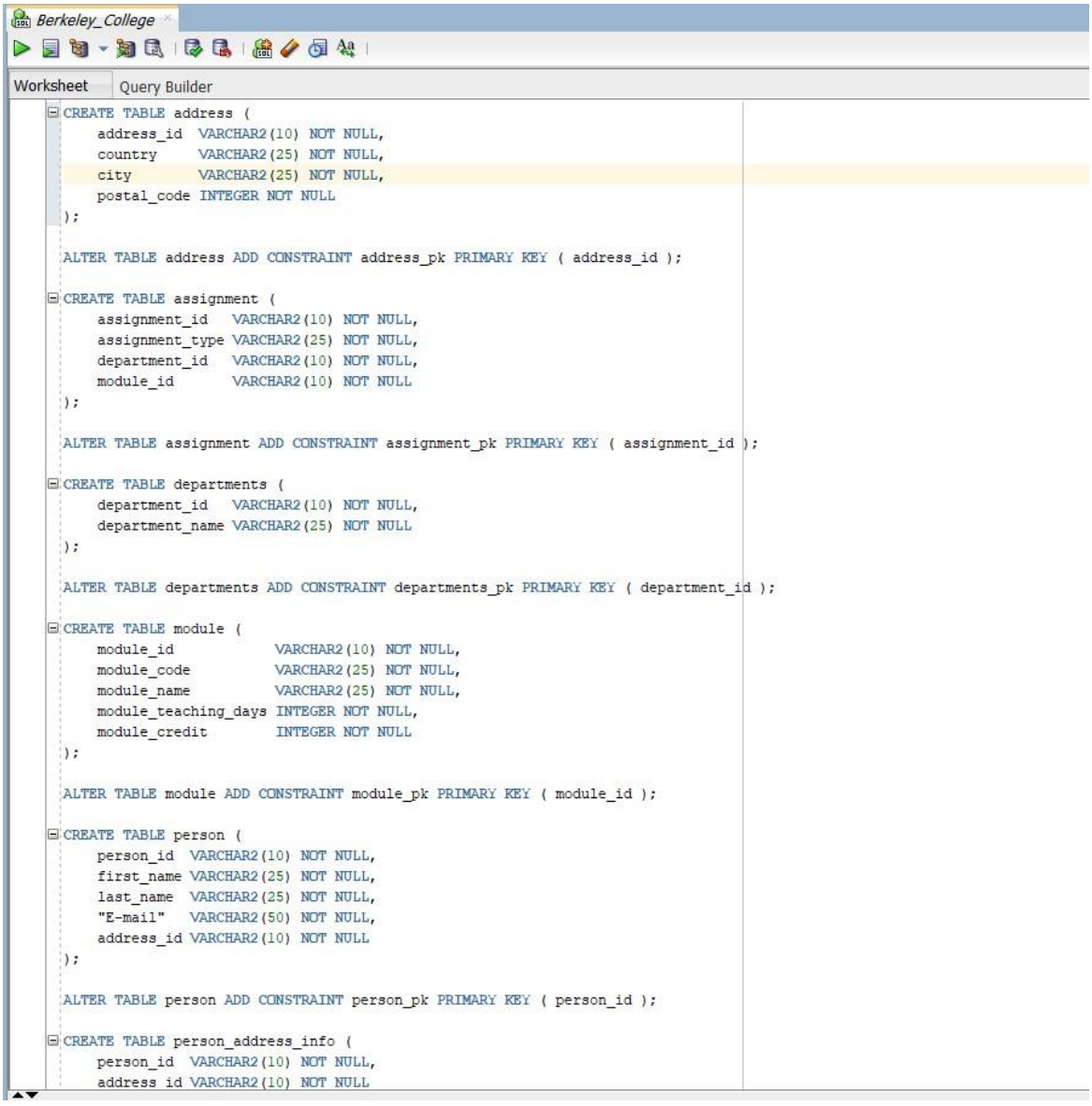


Figure 16: Executing the DDL script for creating tables in SQL Developer

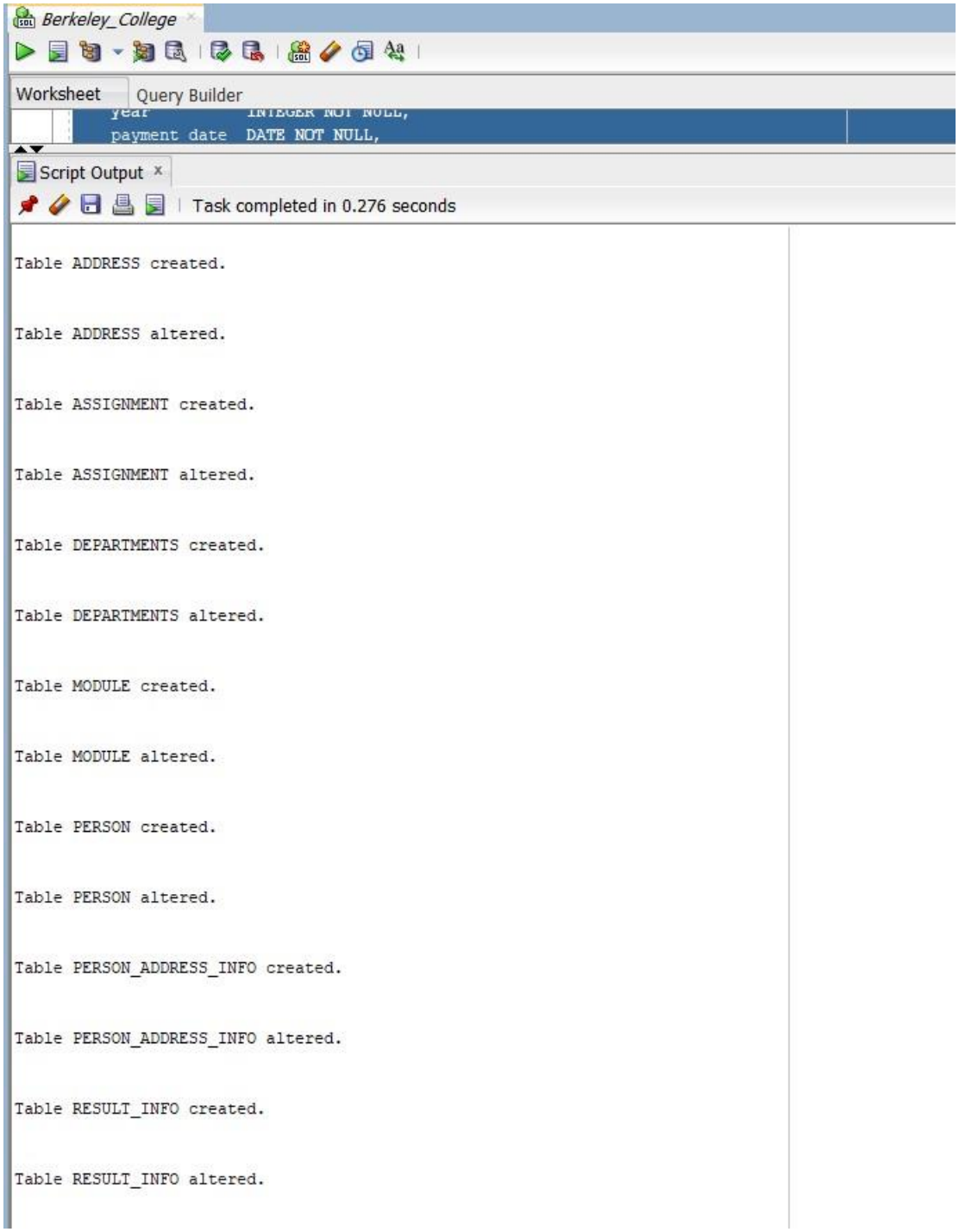


Figure 17: Tables created after executing DDL script in SQL developer

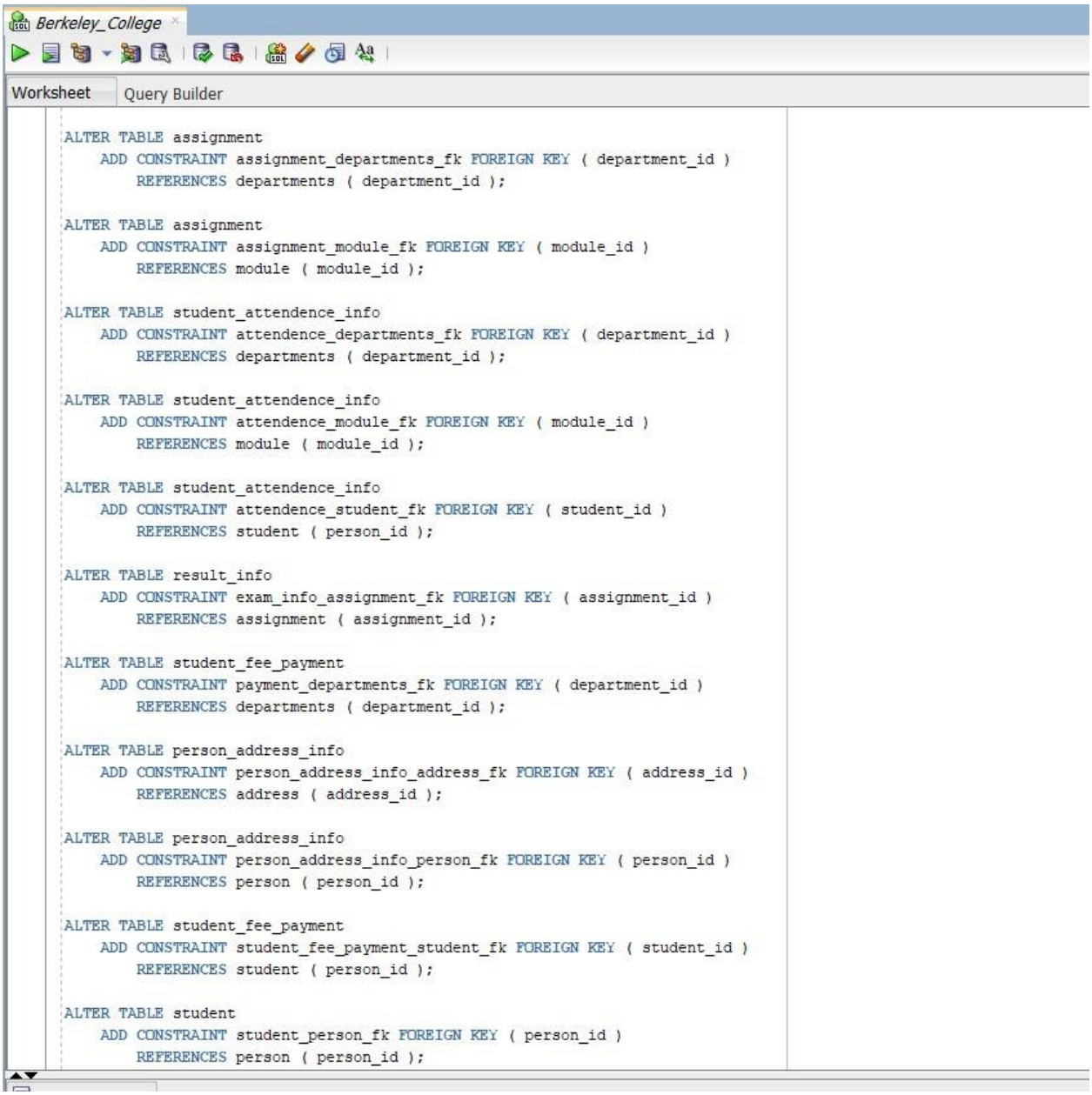


Figure 18: Altering tables with foreign keys in SQL developer

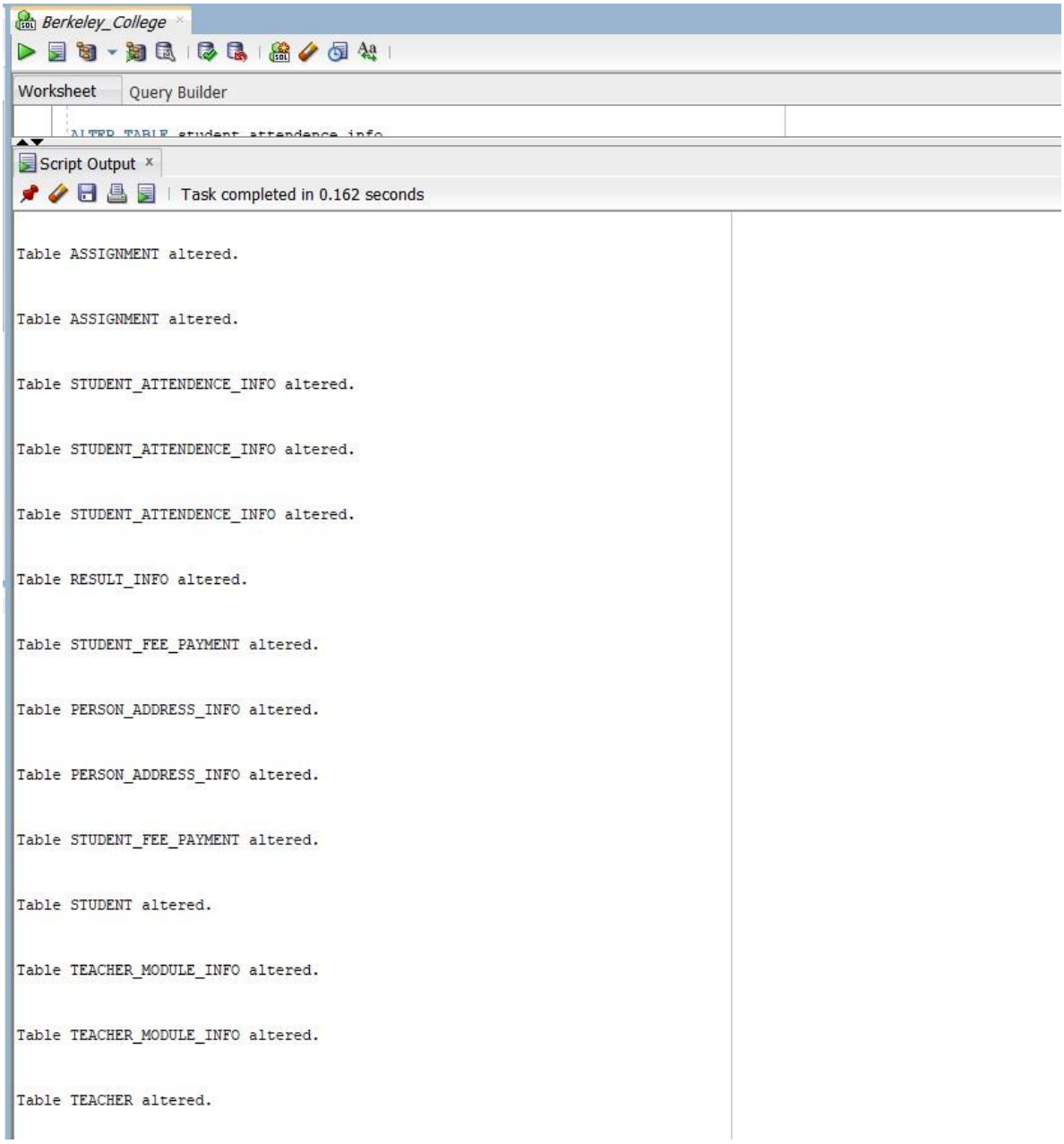


Figure 19 Tables altered with foreign keys in SQL developer

Insert Statements

Insert Statement for Address table

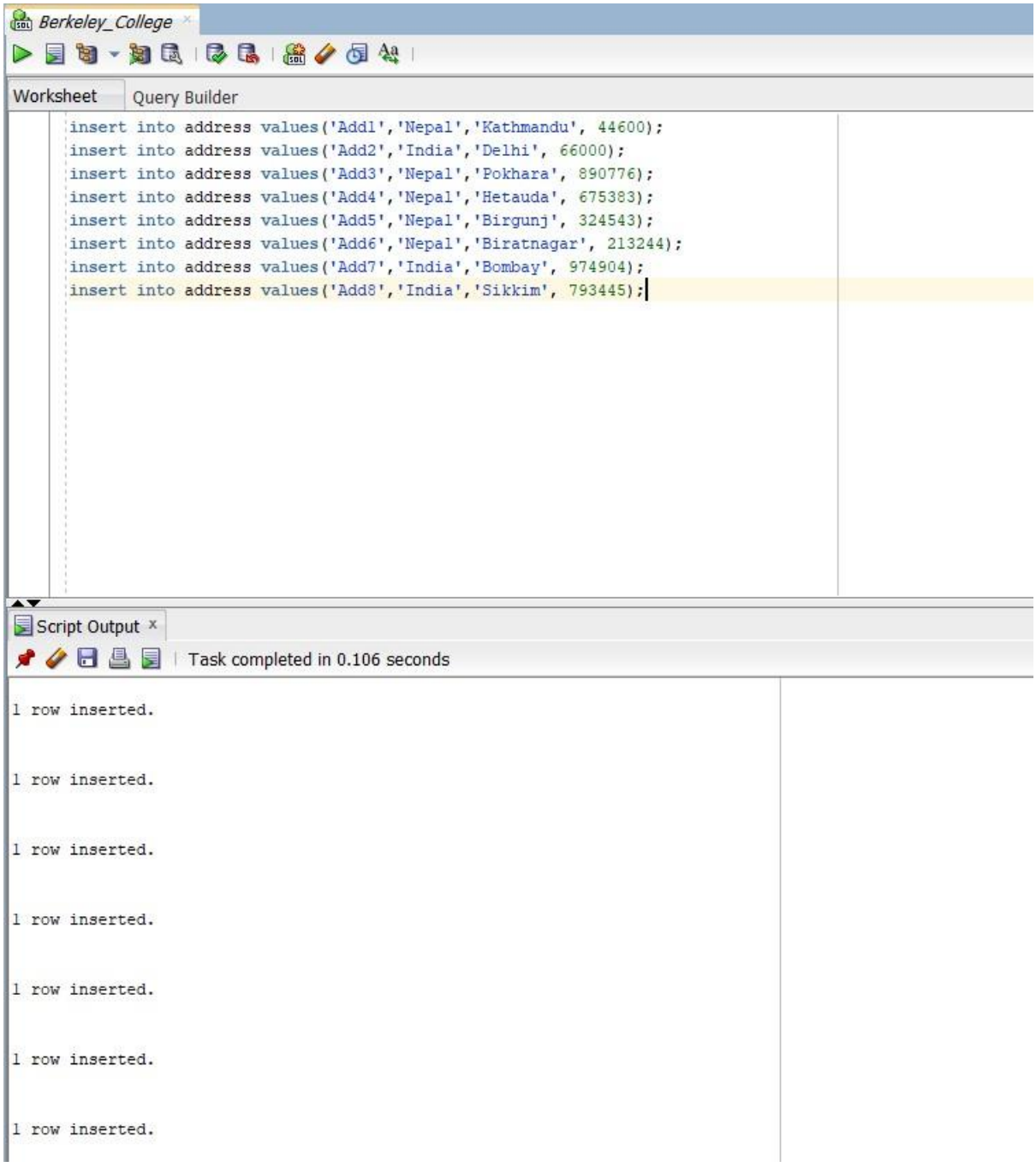


Figure 20: Inserting values in Address Table

Insert Statement for Person table

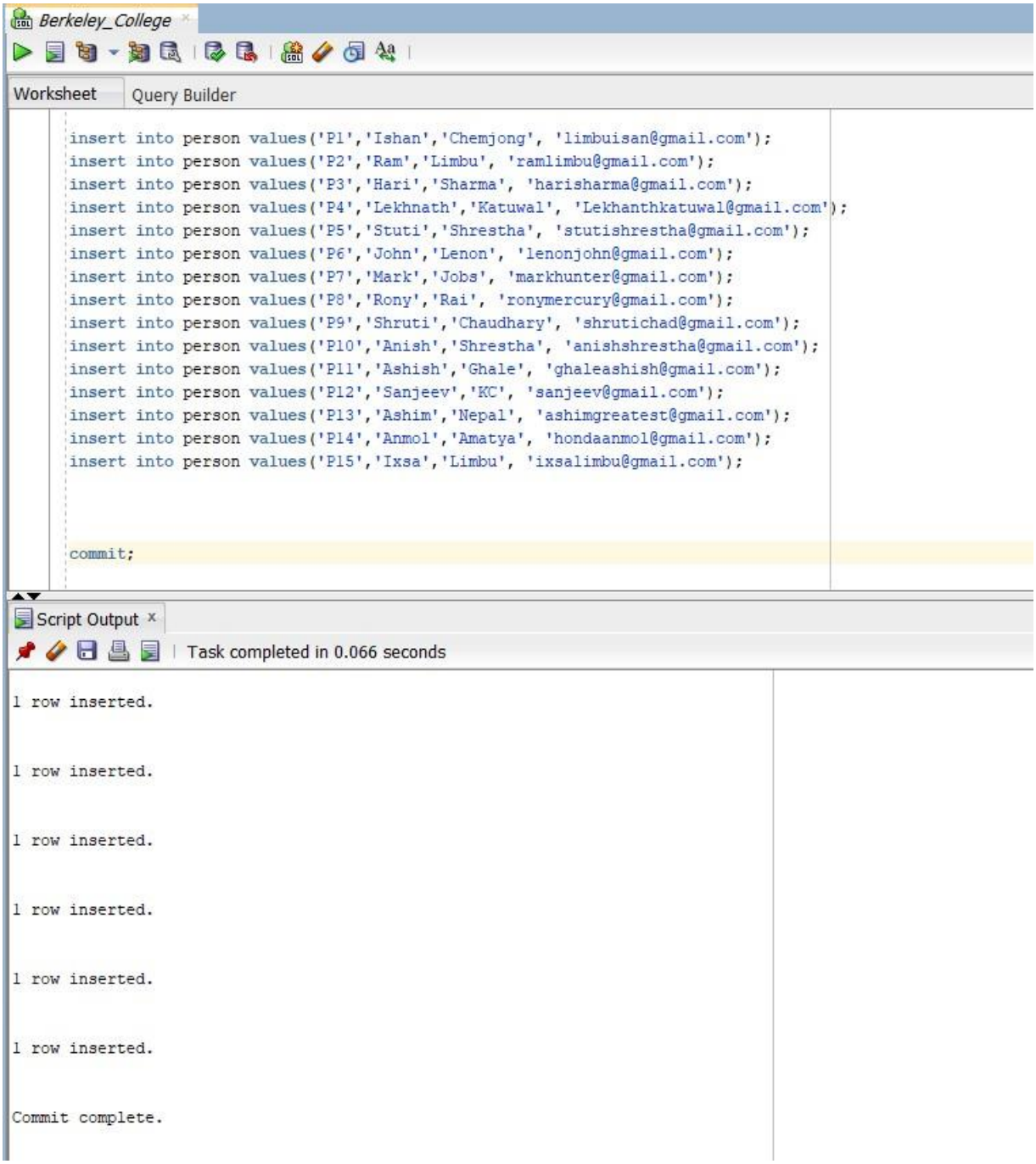


Figure 21: Inserting values in Person Table

Insert Statement for Person_Address_info table

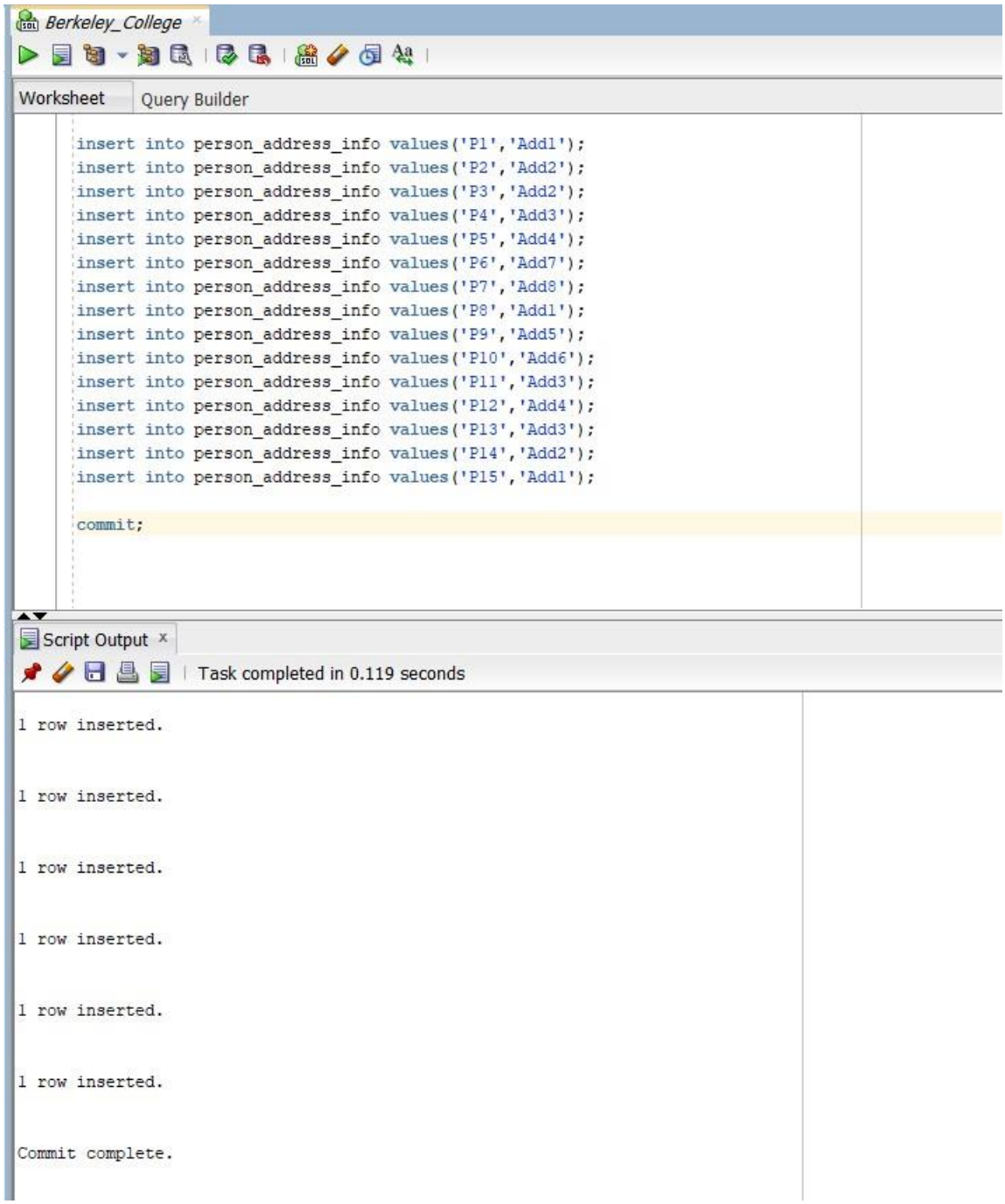


Figure 22: Inserting values in Person_Address_info Table

Insert Statement for Student table

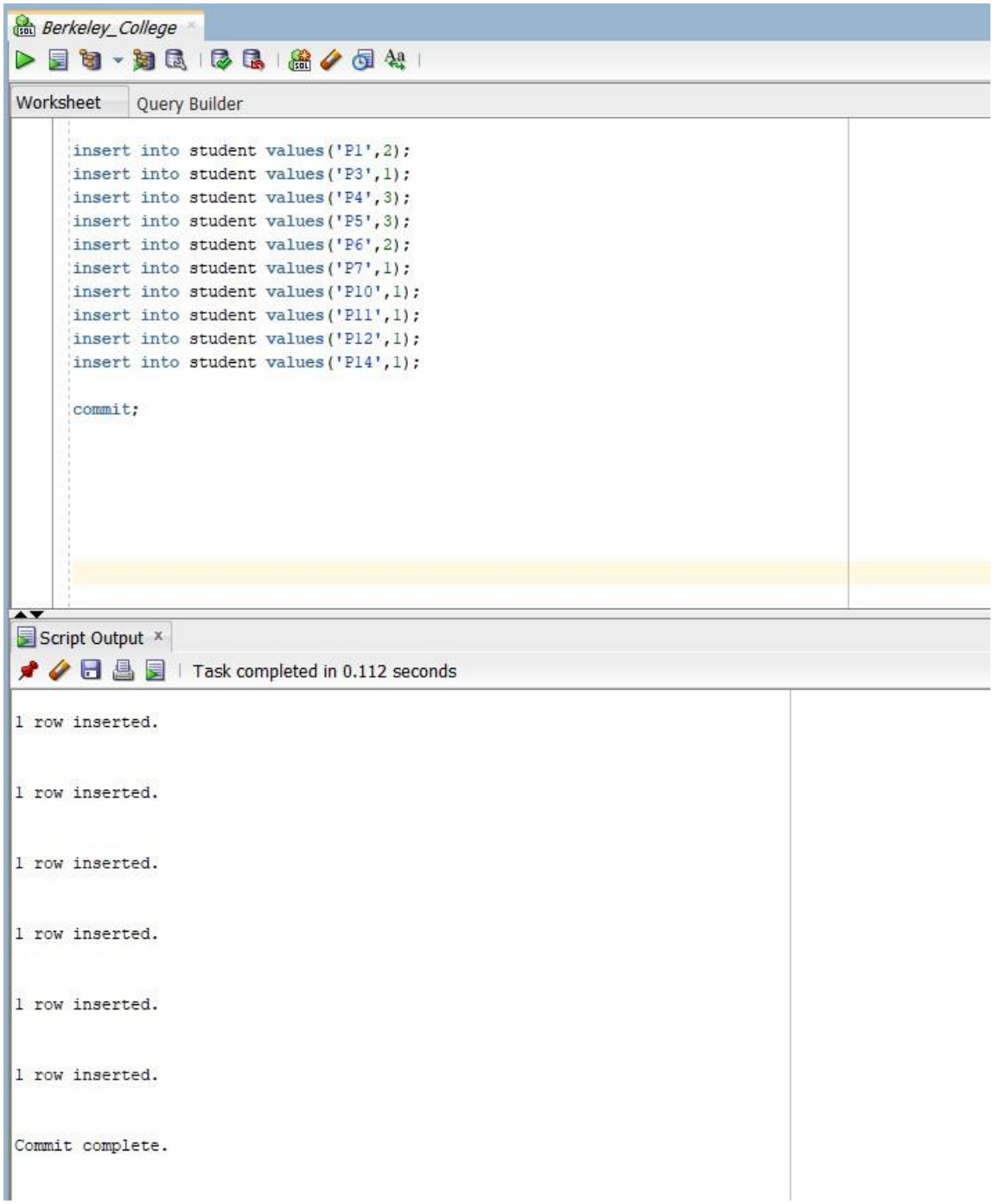


Figure 23: Inserting values in Student Table

Insert Statement for Teacher table

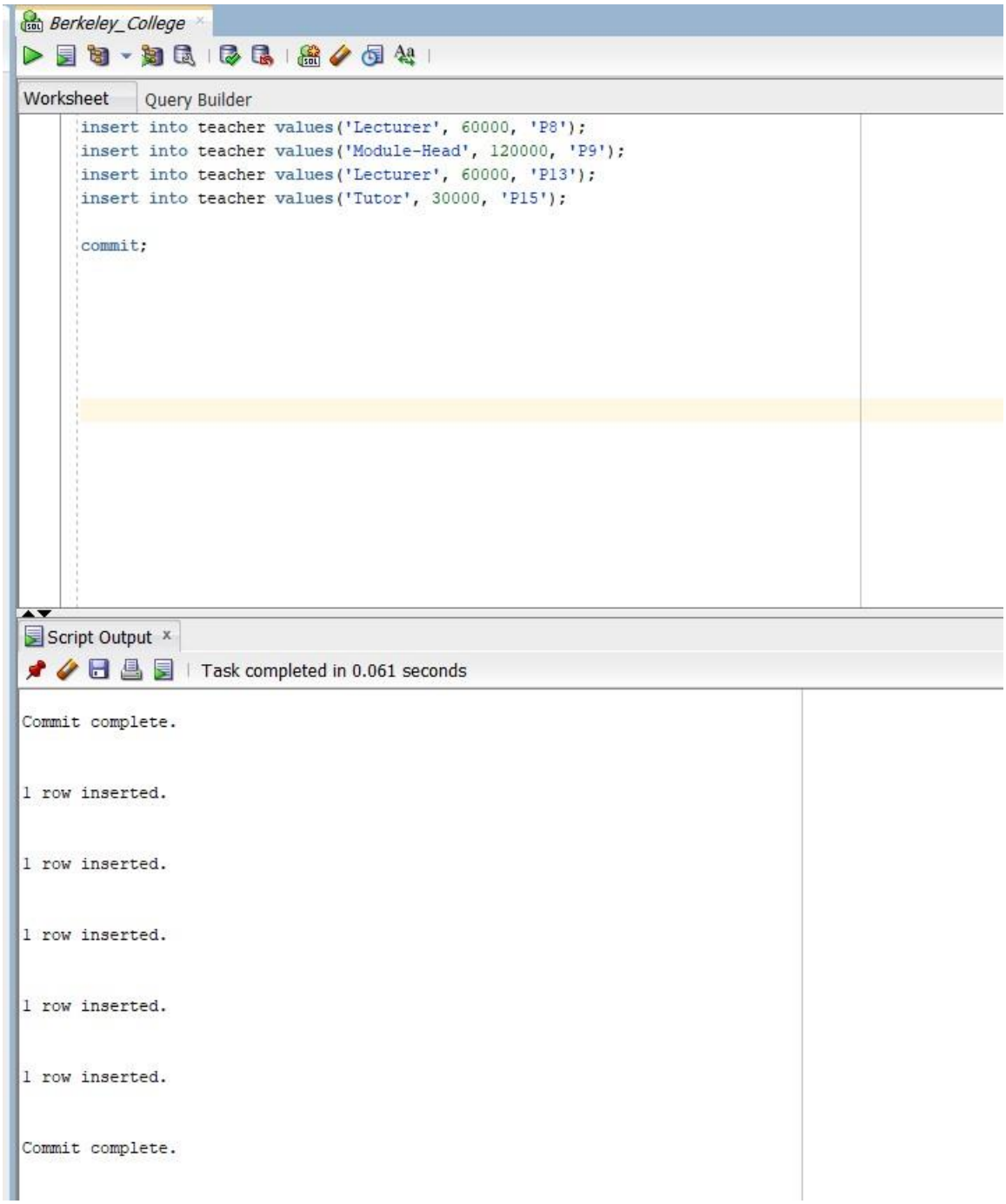


Figure 24: Inserting values in Teacher Table

Insert Statement for Departments table

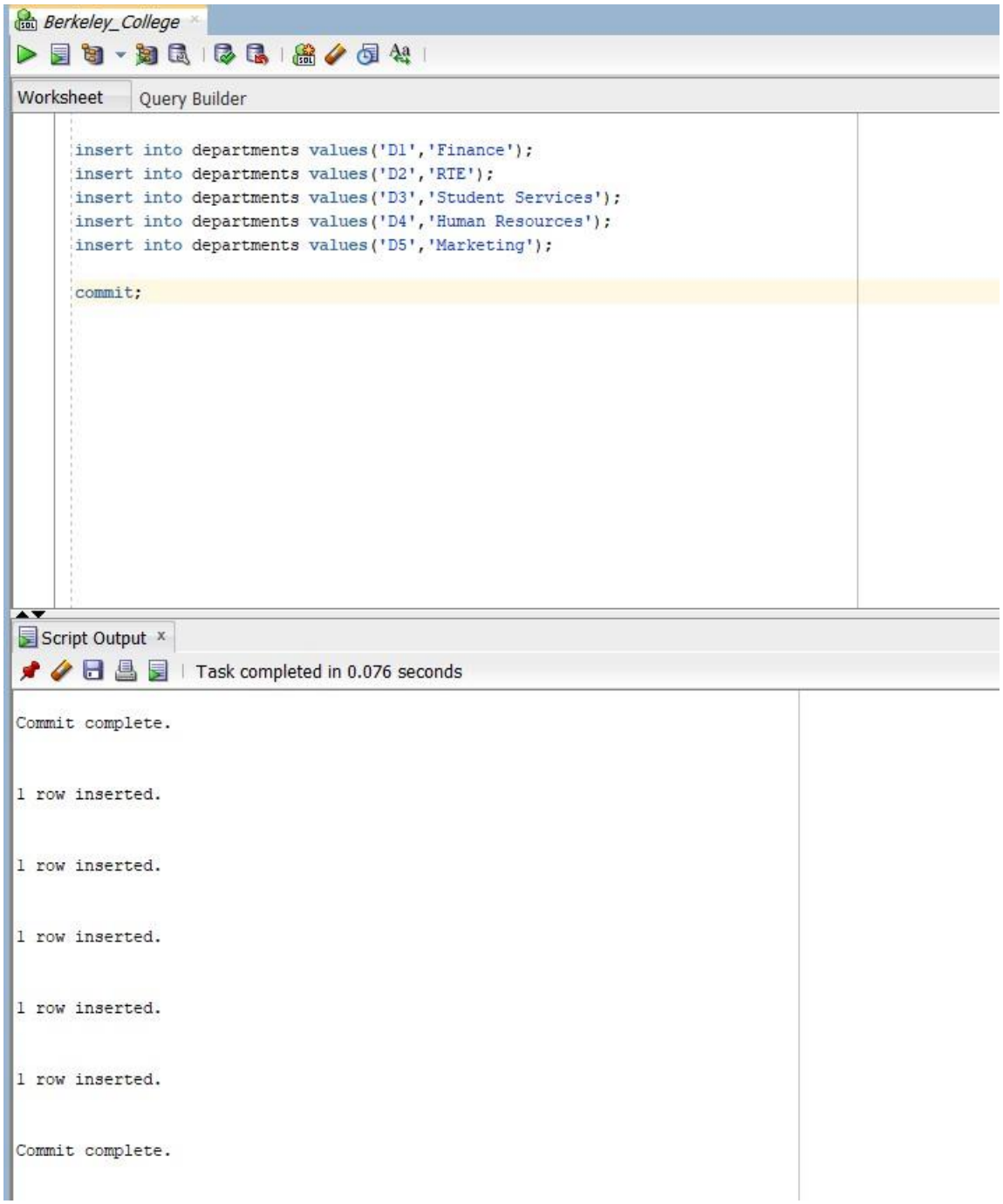


Figure 25: Inserting values in Departments Table

Insert Statement for Module table

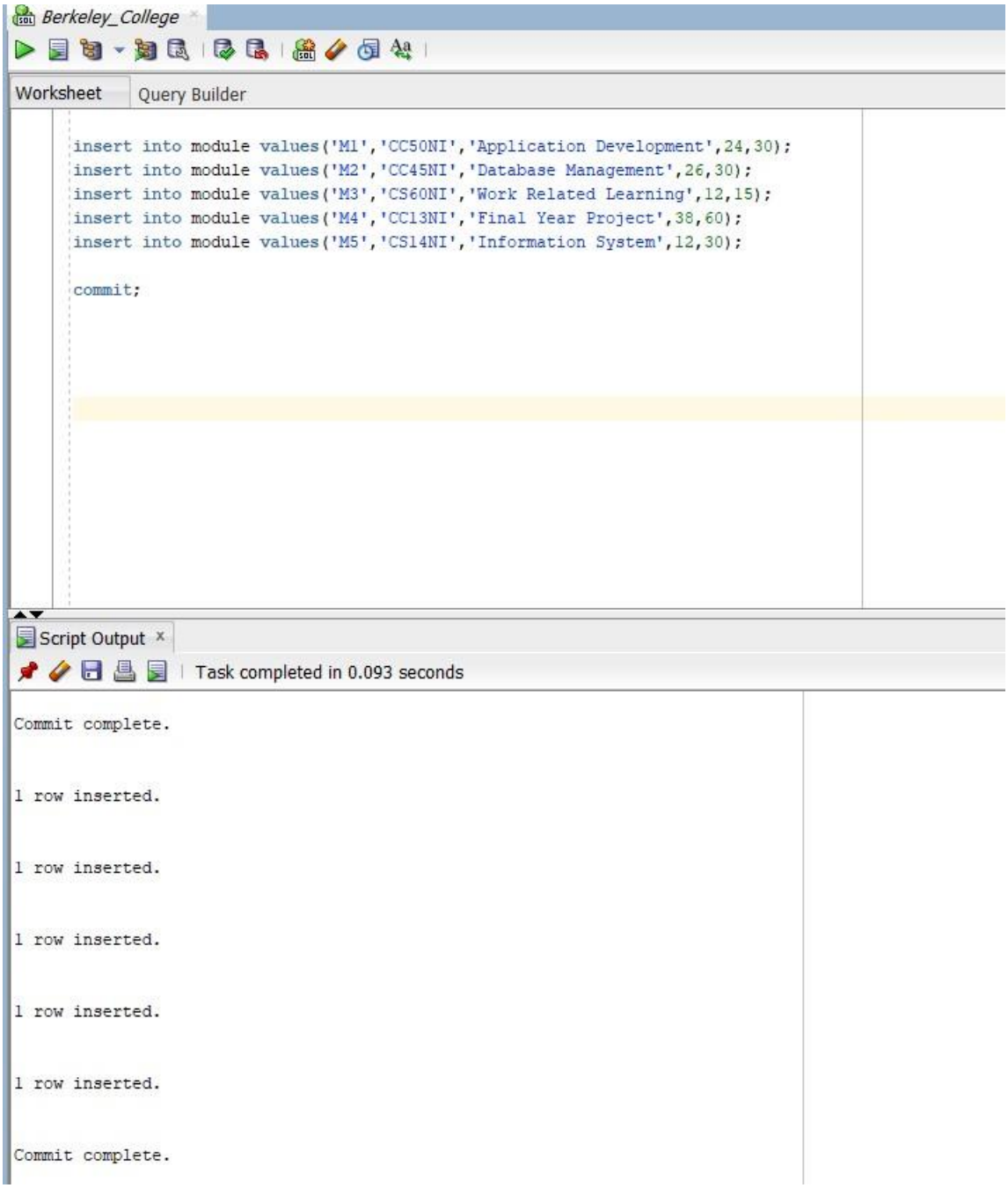


Figure 26: Inserting values in Module Table

Insert Statement for Teacher_Module_Info table

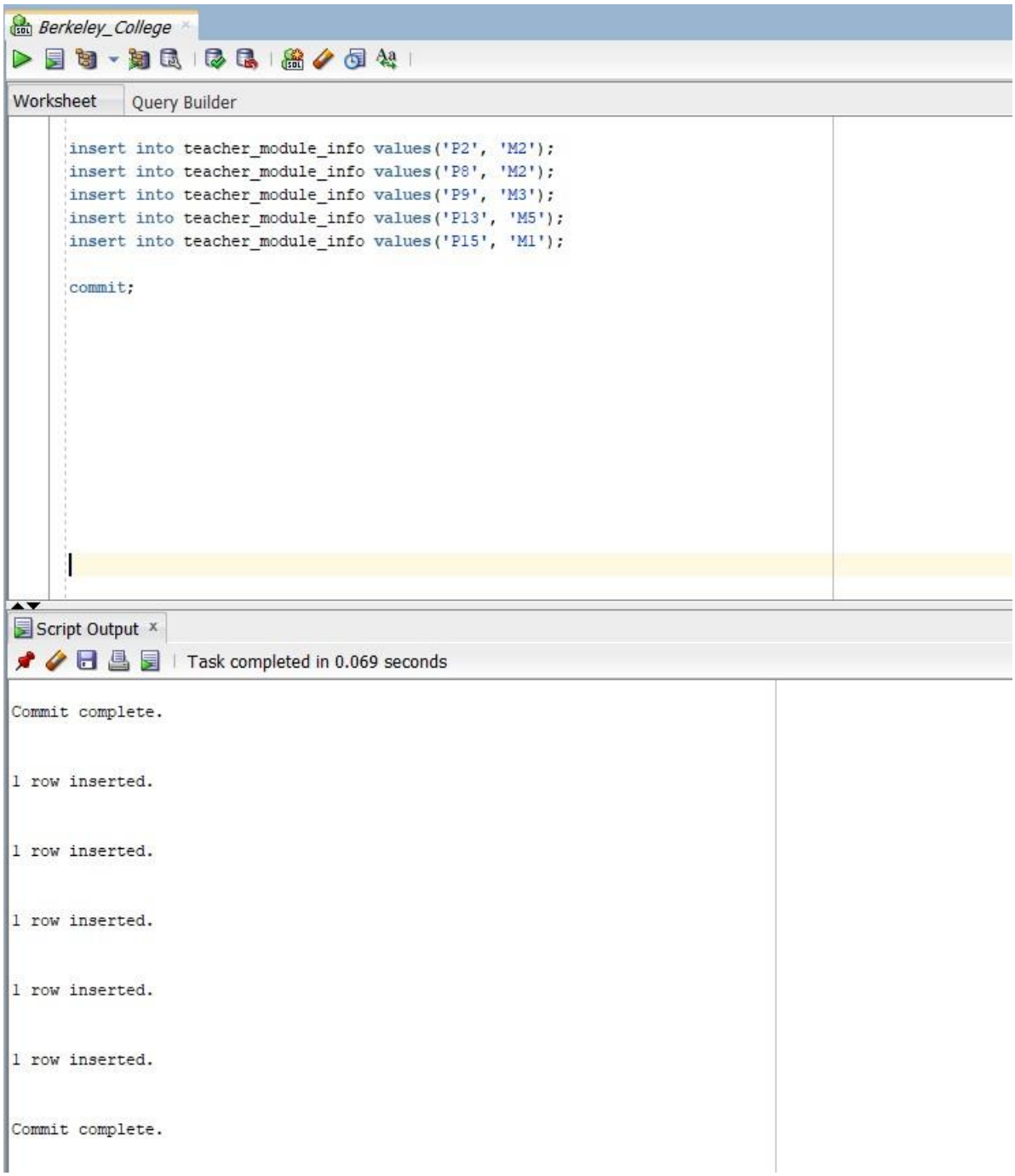


Figure 27: Inserting values in Teacher_Module_info Table

Insert Statement for Assignment table

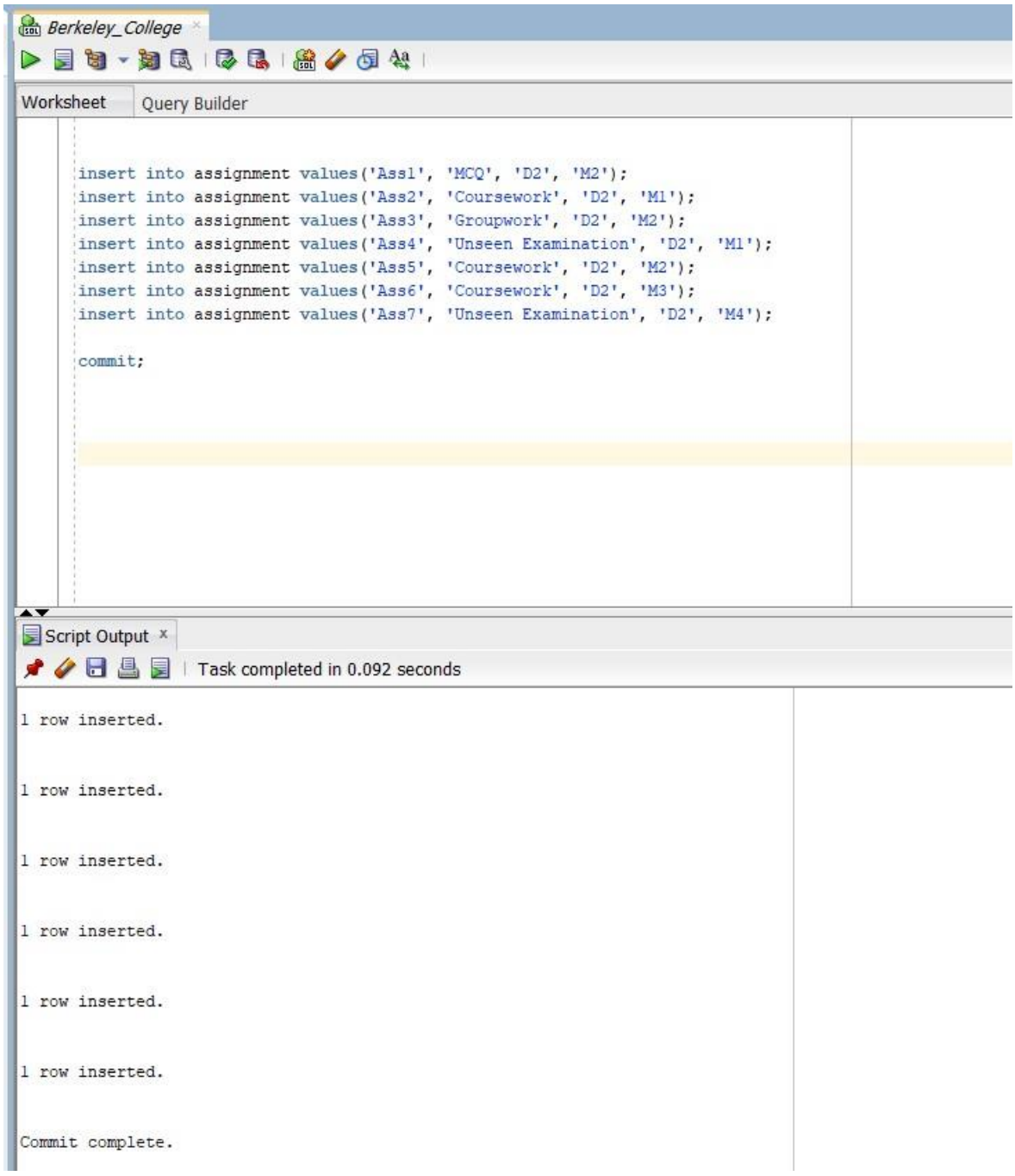


Figure 28: Inserting values in Assignment Table

Insert Statement for Student_Attendance_Info table

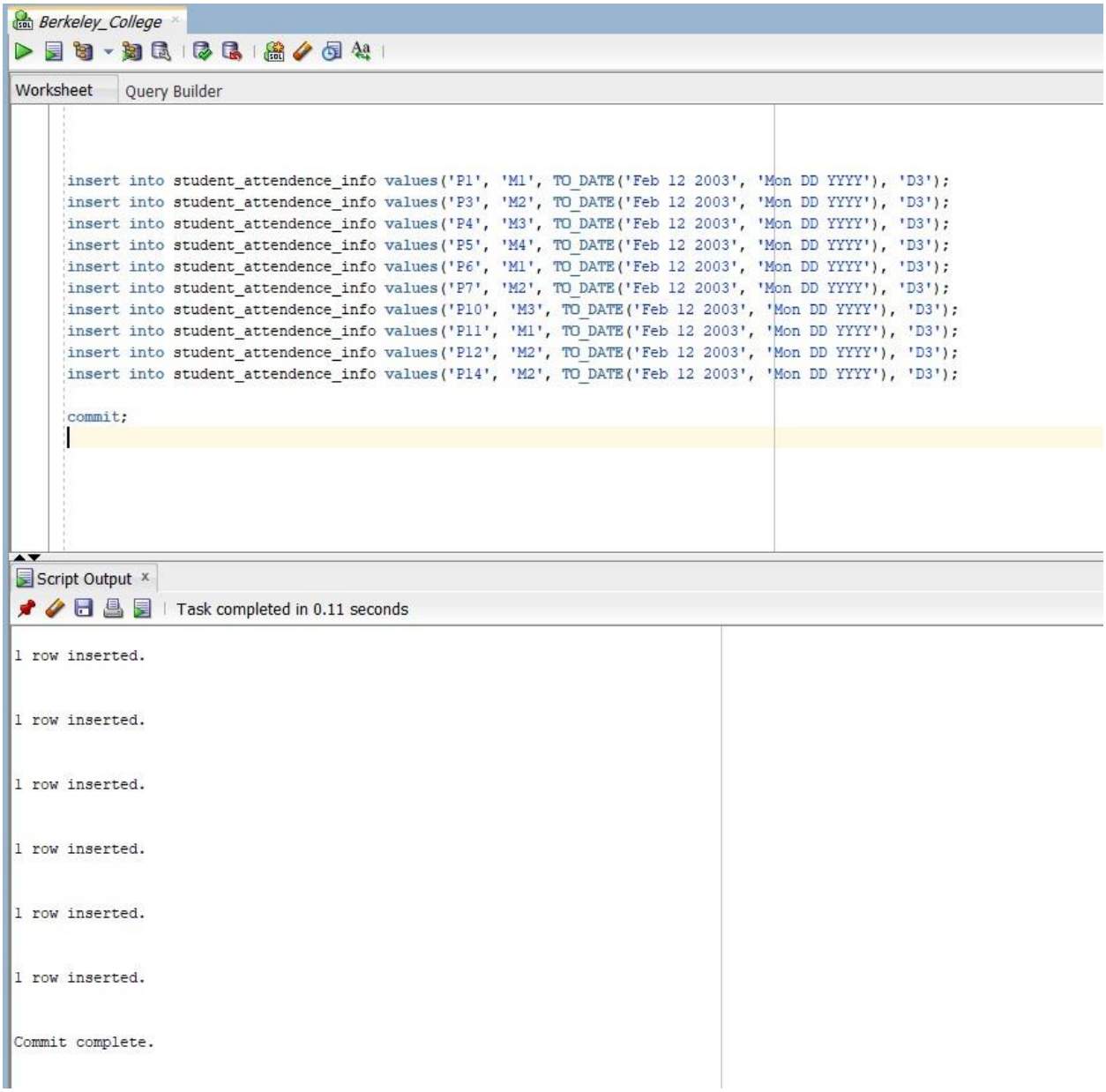


Figure 29: Inserting values in Student_Attendance_info Table

Insert Statement for Student_Fee_Payment table

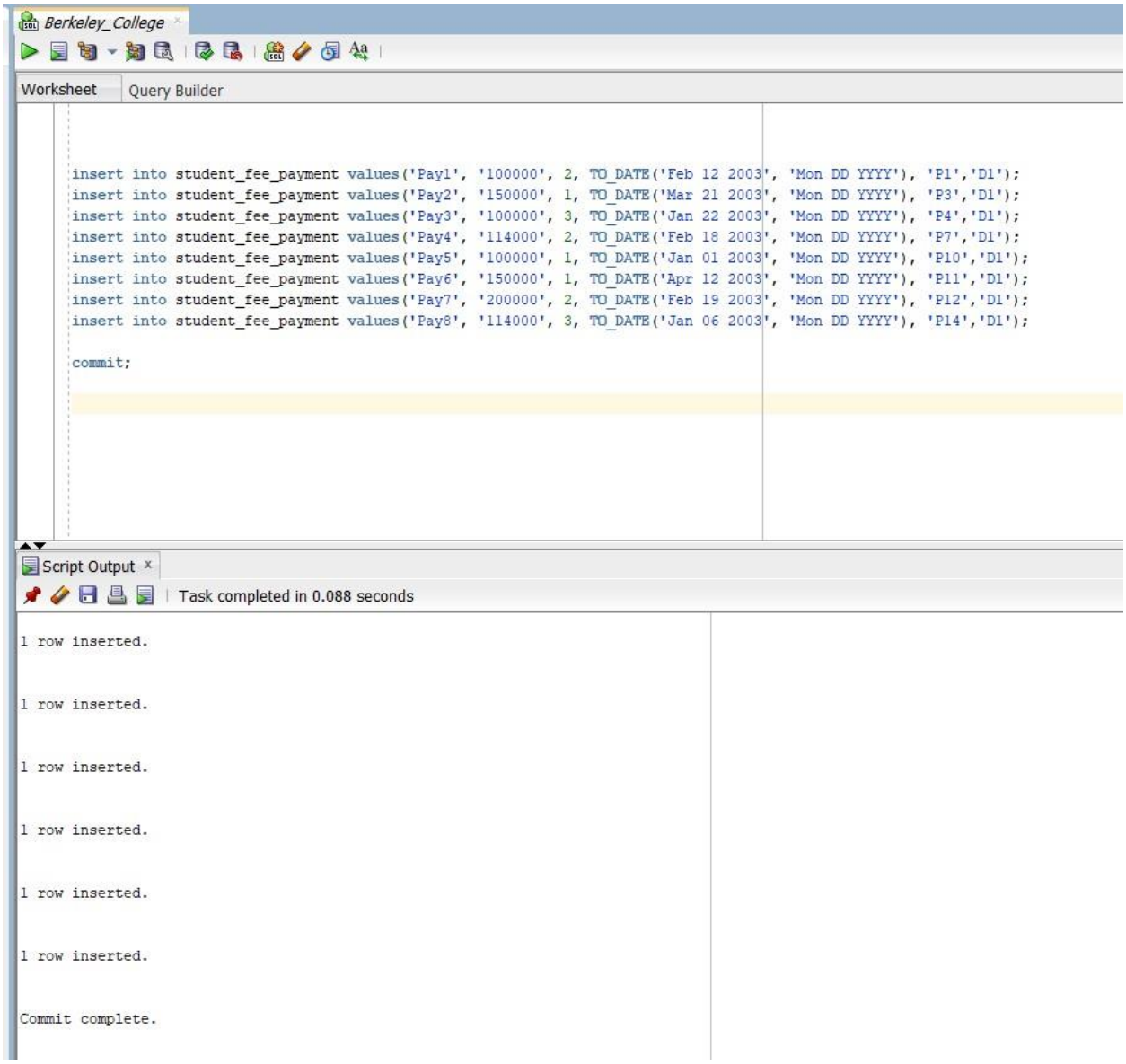


Figure 30: Inserting values in Student_Fee_Payment Table

Insert Statement for Module_Student table

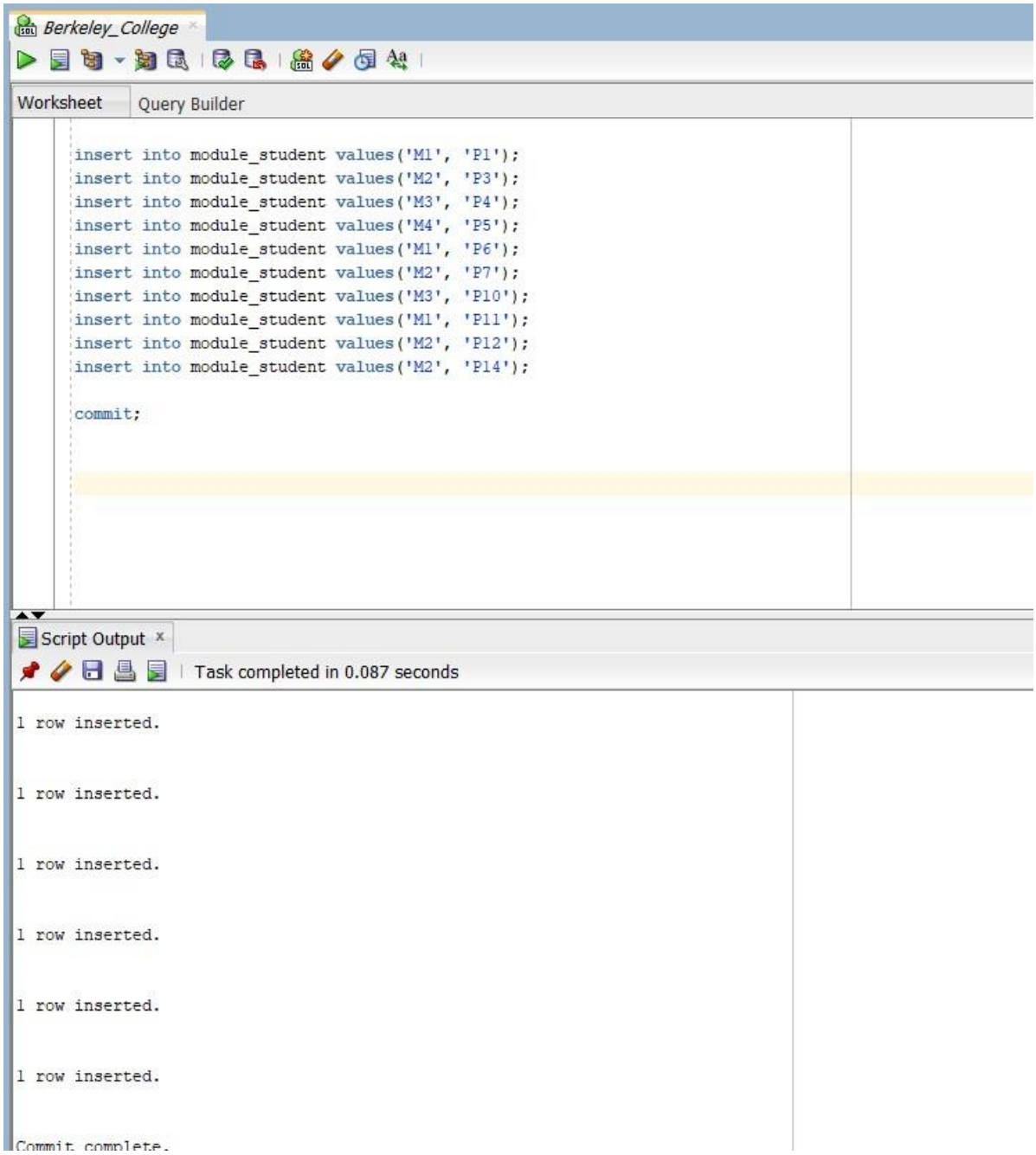


Figure 31: Inserting values in Module_Student Table

Insert Statement for Result_info table

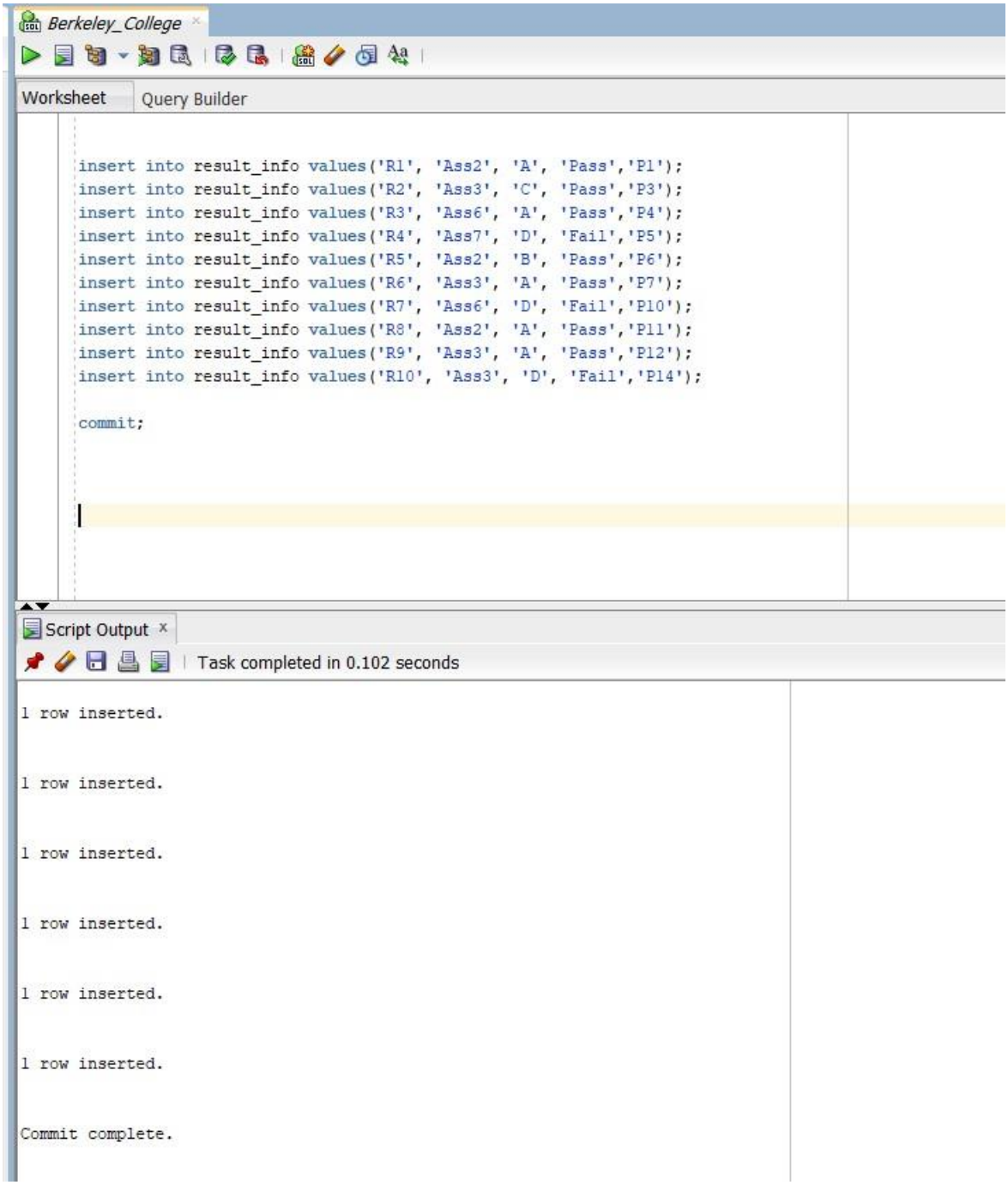
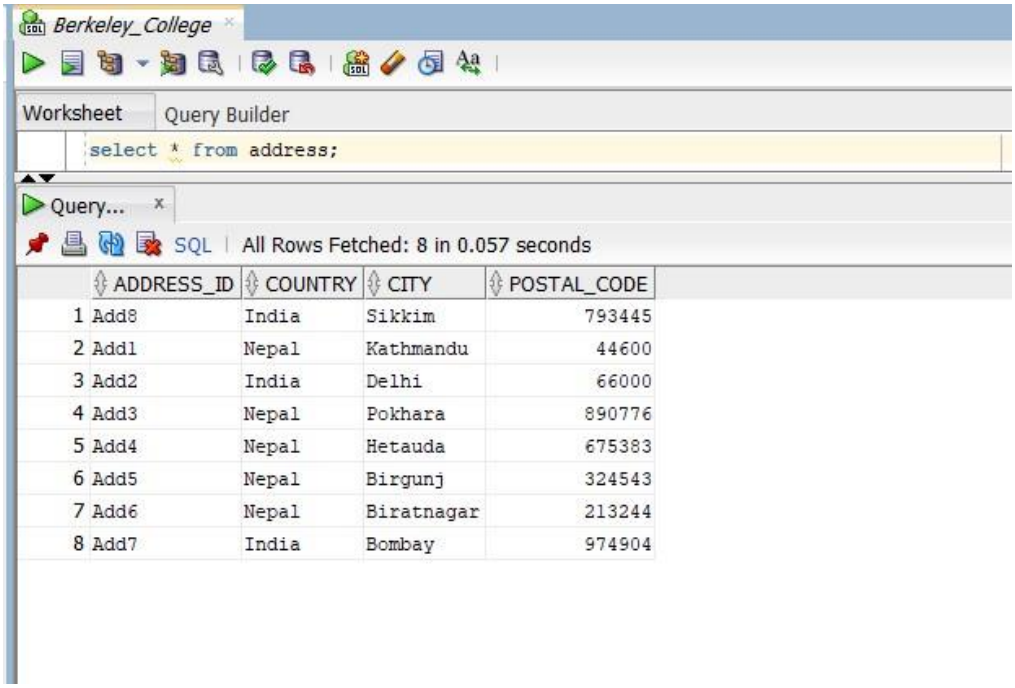


Figure 32: Inserting values in Result_info Table

Select Statements

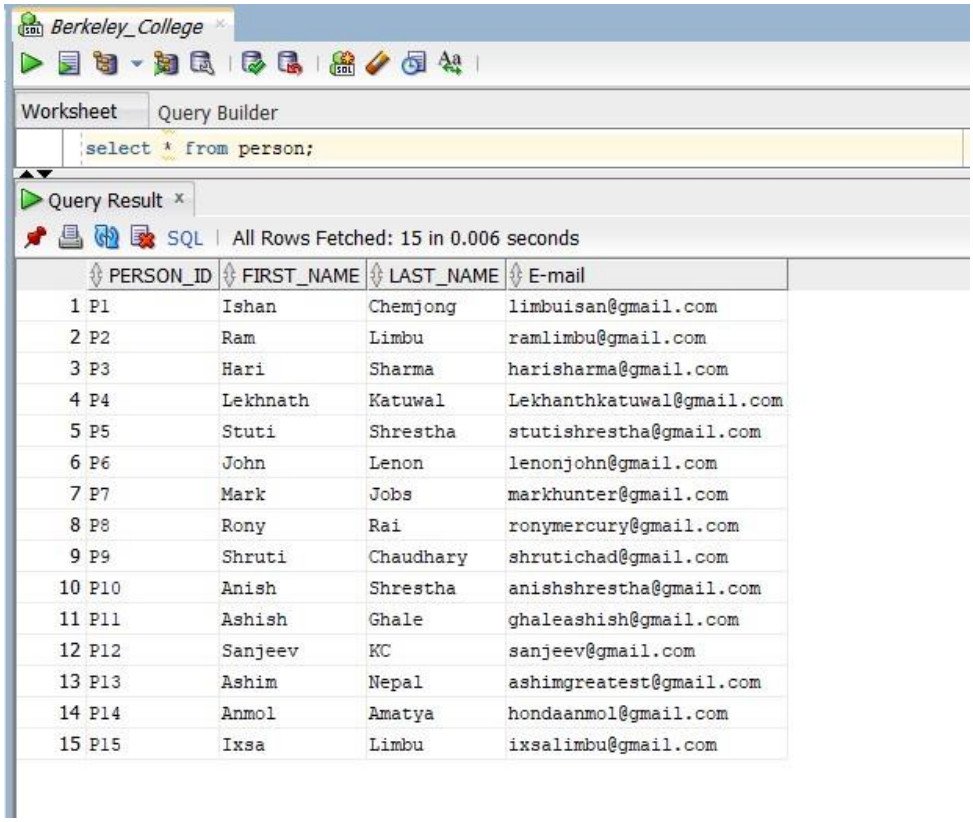
Select Statement for Address table



	ADDRESS_ID	COUNTRY	CITY	POSTAL_CODE
1	Add8	India	Sikkim	793445
2	Add1	Nepal	Kathmandu	44600
3	Add2	India	Delhi	66000
4	Add3	Nepal	Pokhara	890776
5	Add4	Nepal	Hetauda	675383
6	Add5	Nepal	Birgunj	324543
7	Add6	Nepal	Biratnagar	213244
8	Add7	India	Bombay	974904

Figure 33: Select statement in Address Table

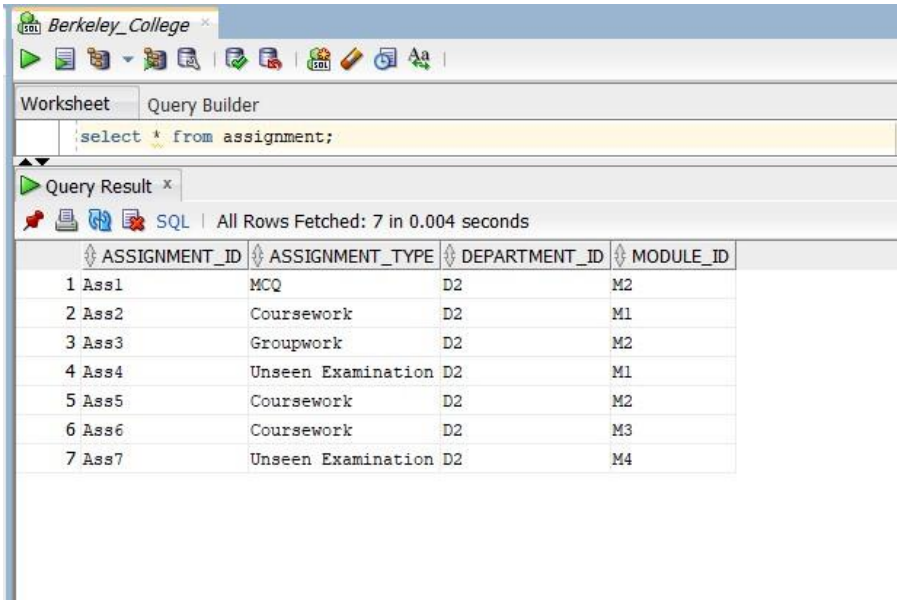
Person table



	PERSON_ID	FIRST_NAME	LAST_NAME	E-mail
1	P1	Ishan	Chemjong	limbuisan@gmail.com
2	P2	Ram	Limbu	ramlimbu@gmail.com
3	P3	Hari	Sharma	harisharma@gmail.com
4	P4	Lekhnath	Katuwal	Lekhanthkatuwal@gmail.com
5	P5	Stuti	Shrestha	stutishrestha@gmail.com
6	P6	John	Lenon	lenonjohn@gmail.com
7	P7	Mark	Jobs	markhunter@gmail.com
8	P8	Rony	Rai	ronymercury@gmail.com
9	P9	Shruti	Chaudhary	shrutichad@gmail.com
10	P10	Anish	Shrestha	anishshrestha@gmail.com
11	P11	Ashish	Ghale	ghaleashish@gmail.com
12	P12	Sanjeev	KC	sanjeev@gmail.com
13	P13	Ashim	Nepal	ashimgreatest@gmail.com
14	P14	Anmol	Amatya	hondaanmol@gmail.com
15	P15	Ixsa	Limbu	ixsalimbu@gmail.com

Figure 34: Select statement in Person Table

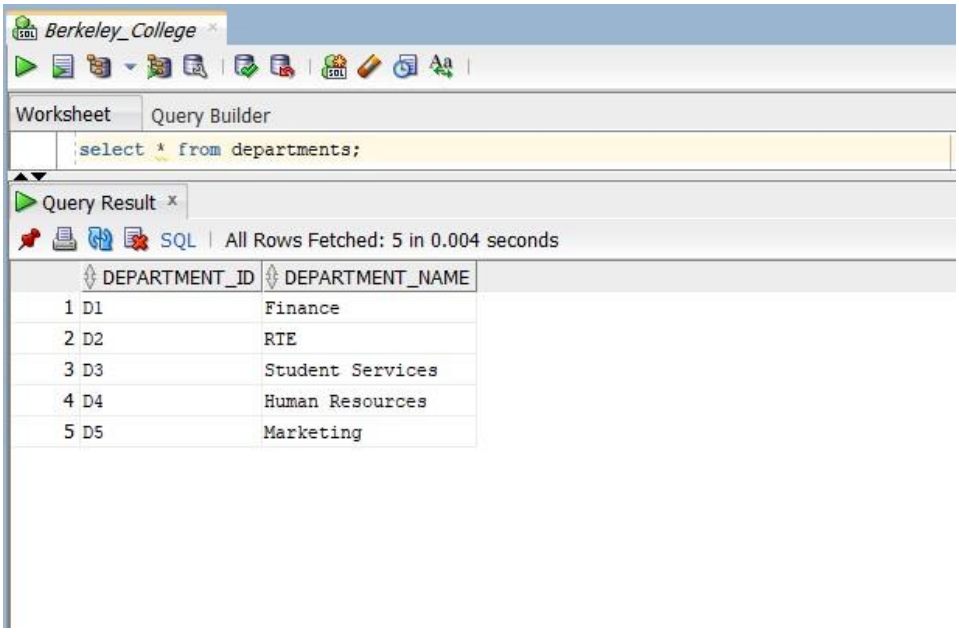
Assignment table



	ASSIGNMENT_ID	ASSIGNMENT_TYPE	DEPARTMENT_ID	MODULE_ID
1	Ass1	MCQ	D2	M2
2	Ass2	Coursework	D2	M1
3	Ass3	Groupwork	D2	M2
4	Ass4	Unseen Examination	D2	M1
5	Ass5	Coursework	D2	M2
6	Ass6	Coursework	D2	M3
7	Ass7	Unseen Examination	D2	M4

Figure 35: Select statement in Assignment Table

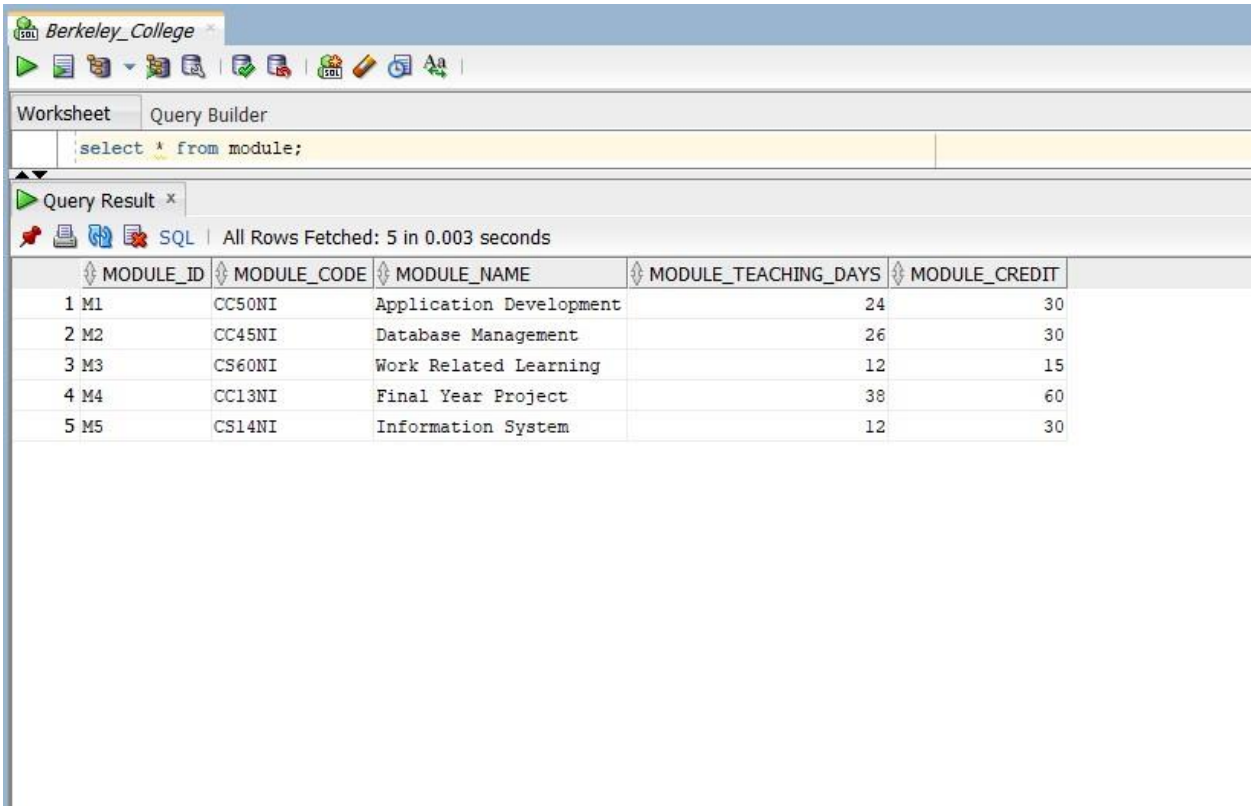
Select Statement for Departments table



	DEPARTMENT_ID	DEPARTMENT_NAME
1	D1	Finance
2	D2	RTE
3	D3	Student Services
4	D4	Human Resources
5	D5	Marketing

Figure 36: Select statement in Departments Table

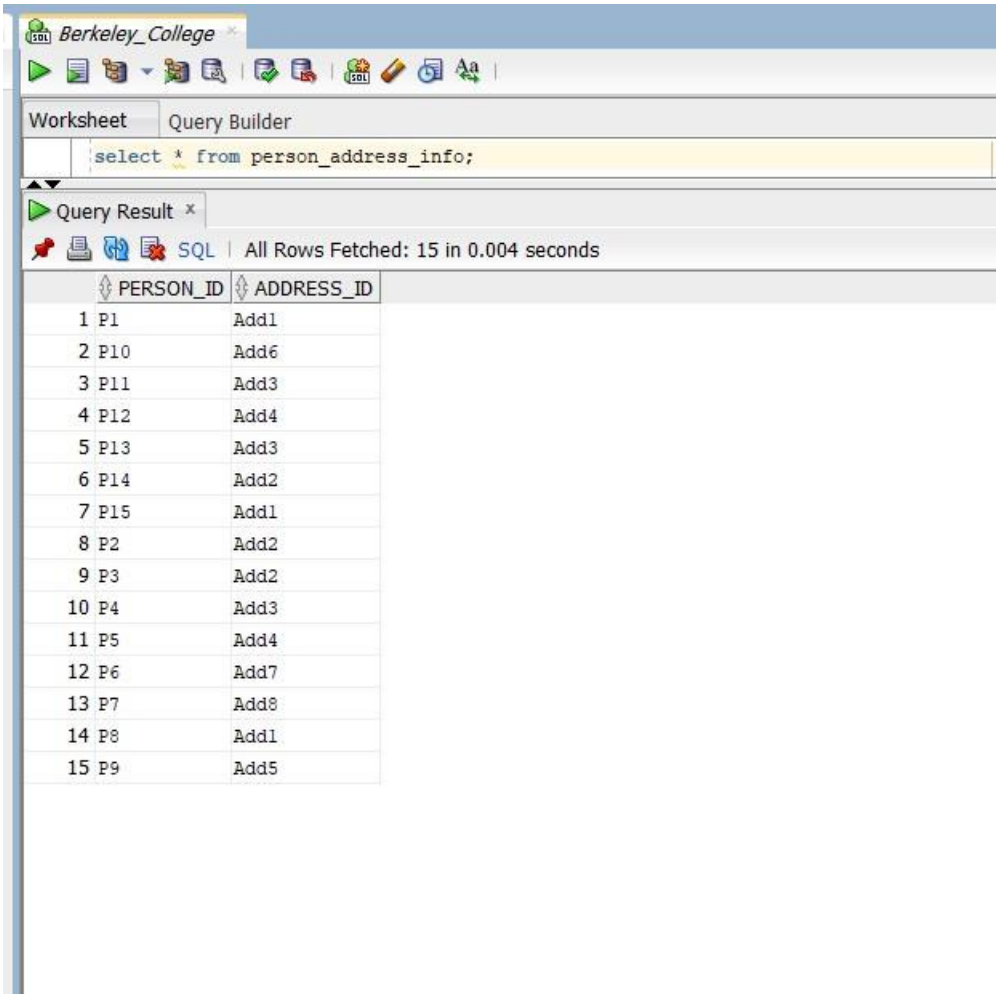
Select Statements for Module table



	MODULE_ID	MODULE_CODE	MODULE_NAME	MODULE_TEACHING_DAYS	MODULE_CREDIT
1	M1	CC50NI	Application Development	24	30
2	M2	CC45NI	Database Management	26	30
3	M3	CS60NI	Work Related Learning	12	15
4	M4	CC13NI	Final Year Project	38	60
5	M5	CS14NI	Information System	12	30

Figure 37: Select statement in Module Table

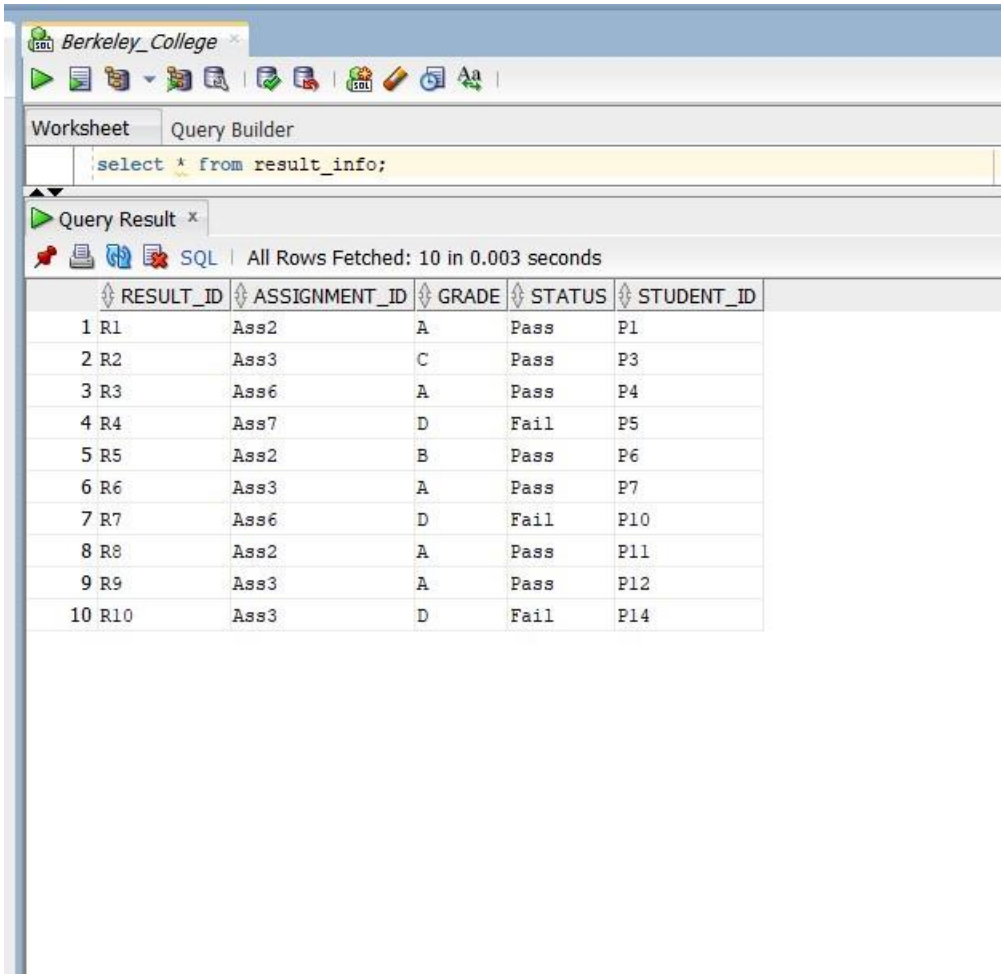
Select Statement for Person_Address_info table



	PERSON_ID	ADDRESS_ID
1	P1	Add1
2	P10	Add6
3	P11	Add3
4	P12	Add4
5	P13	Add3
6	P14	Add2
7	P15	Add1
8	P2	Add2
9	P3	Add2
10	P4	Add3
11	P5	Add4
12	P6	Add7
13	P7	Add8
14	P8	Add1
15	P9	Add5

Figure 38: Select statement in Person_Address_info Table

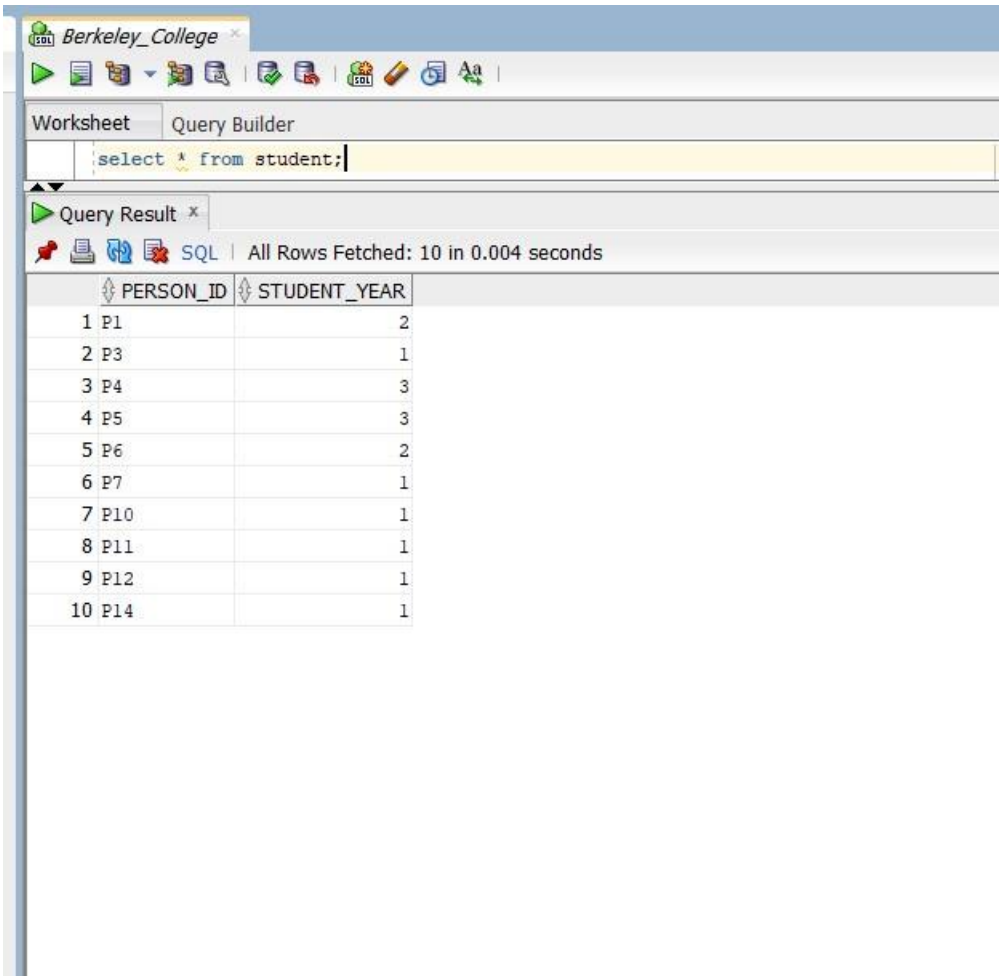
Select Statement for Result_info table



	RESULT_ID	ASSIGNMENT_ID	GRADE	STATUS	STUDENT_ID
1	R1	Ass2	A	Pass	P1
2	R2	Ass3	C	Pass	P3
3	R3	Ass6	A	Pass	P4
4	R4	Ass7	D	Fail	P5
5	R5	Ass2	B	Pass	P6
6	R6	Ass3	A	Pass	P7
7	R7	Ass6	D	Fail	P10
8	R8	Ass2	A	Pass	P11
9	R9	Ass3	A	Pass	P12
10	R10	Ass3	D	Fail	P14

Figure 39: Select statement in Result_info Table

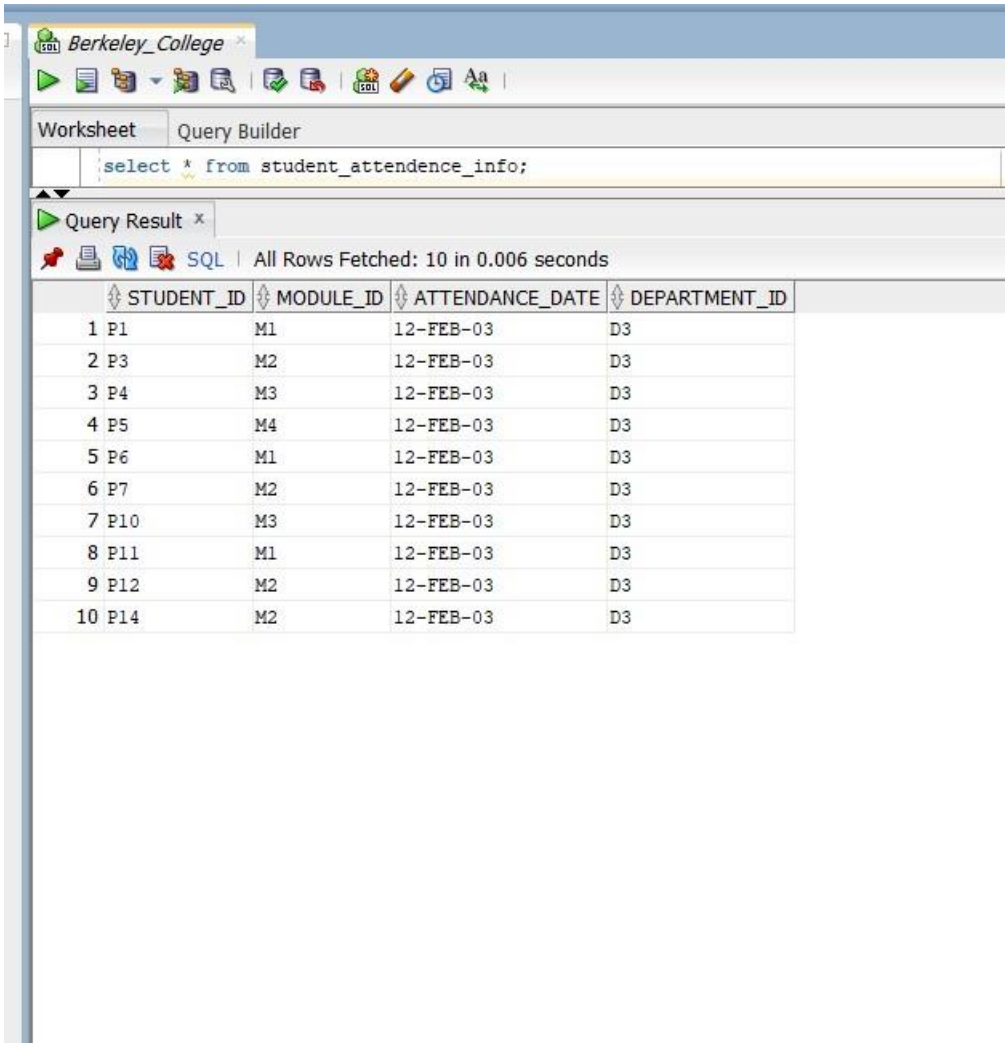
Select Statement for Student table



	PERSON_ID	STUDENT_YEAR
1	P1	2
2	P3	1
3	P4	3
4	P5	3
5	P6	2
6	P7	1
7	P10	1
8	P11	1
9	P12	1
10	P14	1

Figure 40: Select statement in Student Table

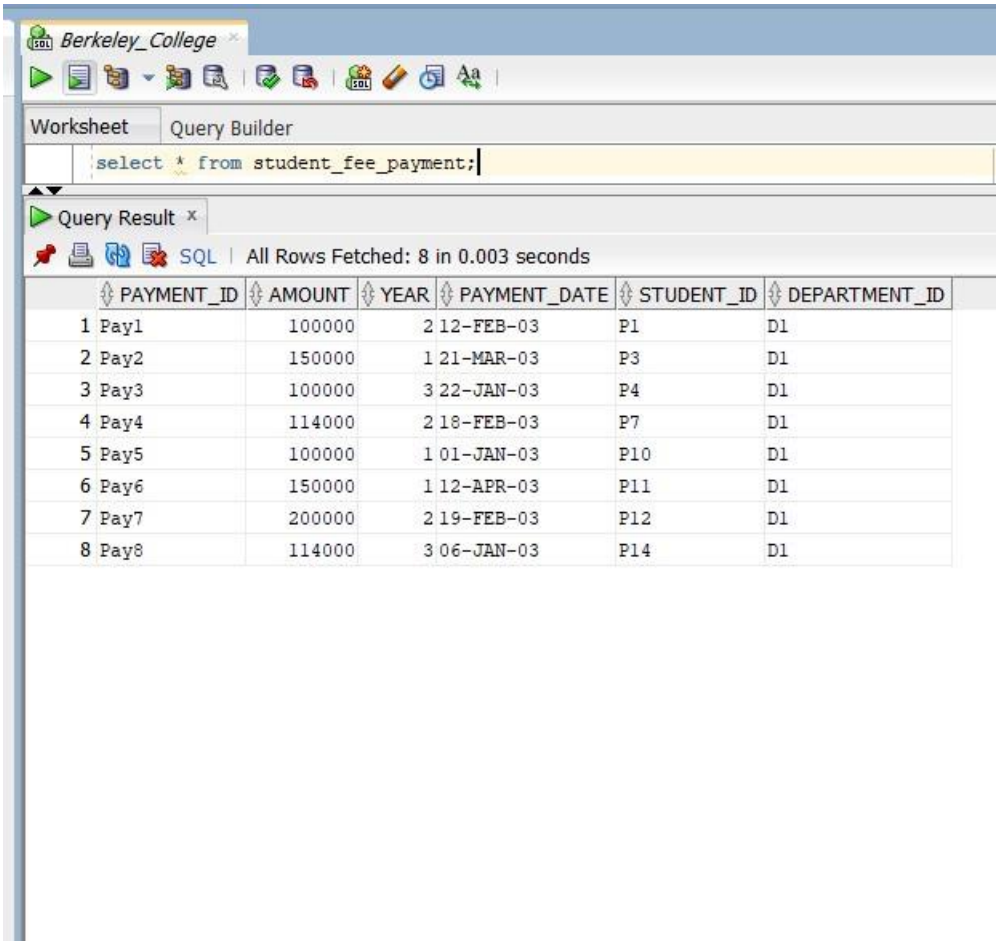
Select Statement for Student_Attendance_info table



	STUDENT_ID	MODULE_ID	ATTENDANCE_DATE	DEPARTMENT_ID
1	P1	M1	12-FEB-03	D3
2	P3	M2	12-FEB-03	D3
3	P4	M3	12-FEB-03	D3
4	P5	M4	12-FEB-03	D3
5	P6	M1	12-FEB-03	D3
6	P7	M2	12-FEB-03	D3
7	P10	M3	12-FEB-03	D3
8	P11	M1	12-FEB-03	D3
9	P12	M2	12-FEB-03	D3
10	P14	M2	12-FEB-03	D3

Figure 41: Select statement in Student_Attendance_info Table

Select Statement for Student_Fee_Payment table



	PAYMENT_ID	AMOUNT	YEAR	PAYMENT_DATE	STUDENT_ID	DEPARTMENT_ID
1	Pay1	100000	2	12-FEB-03	P1	D1
2	Pay2	150000	1	21-MAR-03	P3	D1
3	Pay3	100000	3	22-JAN-03	P4	D1
4	Pay4	114000	2	18-FEB-03	P7	D1
5	Pay5	100000	1	01-JAN-03	P10	D1
6	Pay6	150000	1	12-APR-03	P11	D1
7	Pay7	200000	2	19-FEB-03	P12	D1
8	Pay8	114000	3	06-JAN-03	P14	D1

Figure 42: Select statement in Student_Fee_Payment Table

Select Statement for Teacher table

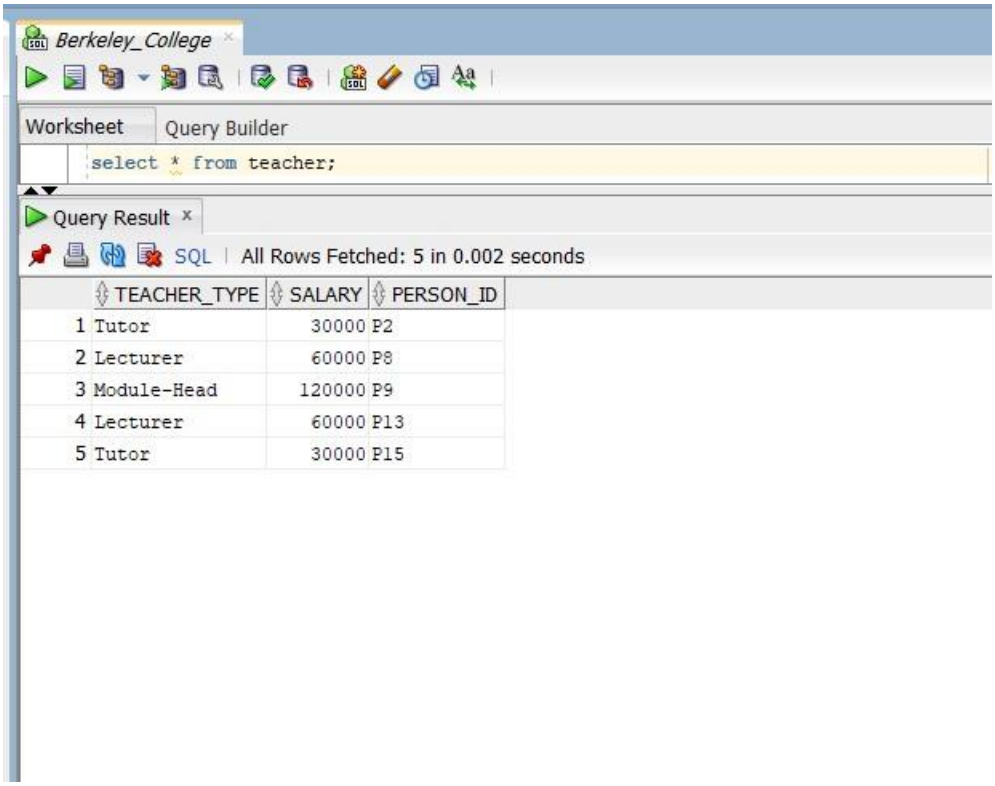


Figure 43: Select statement in Teacher Table

Select Statement for Teacher_Module_info table

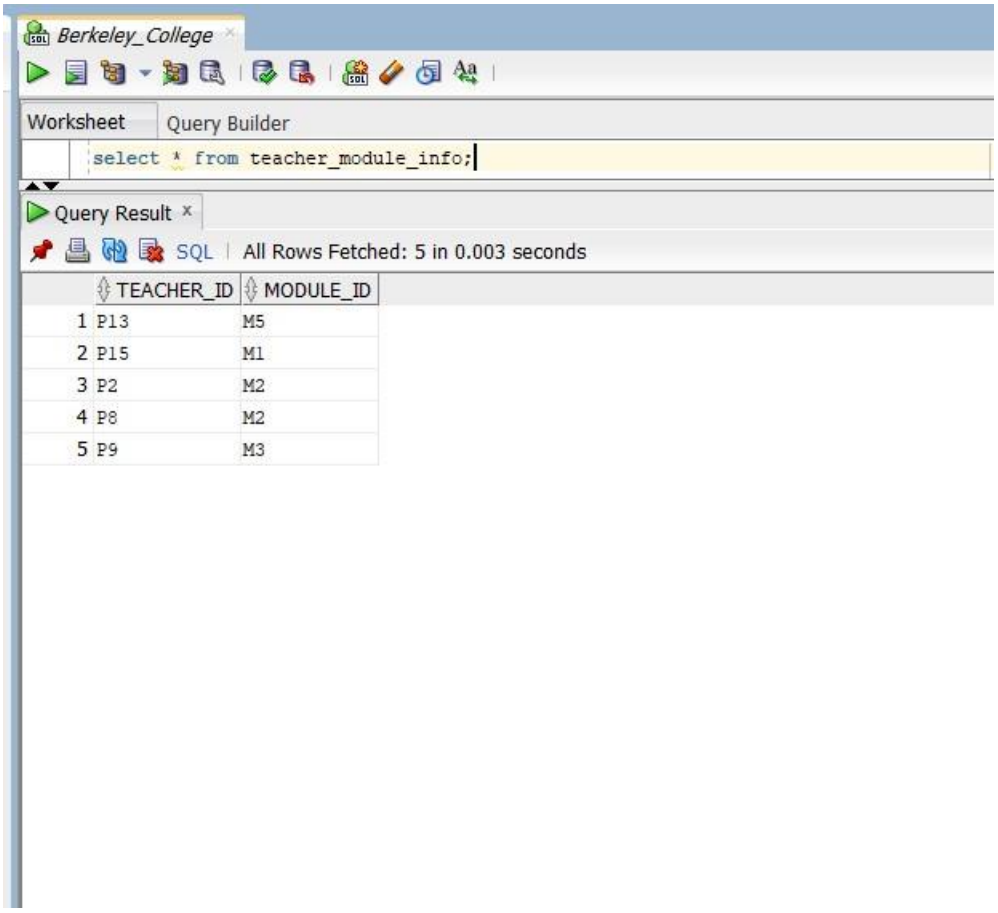


Figure 44: Select statement in Teacher_Module_info Table

Select Statement for Module_Student table

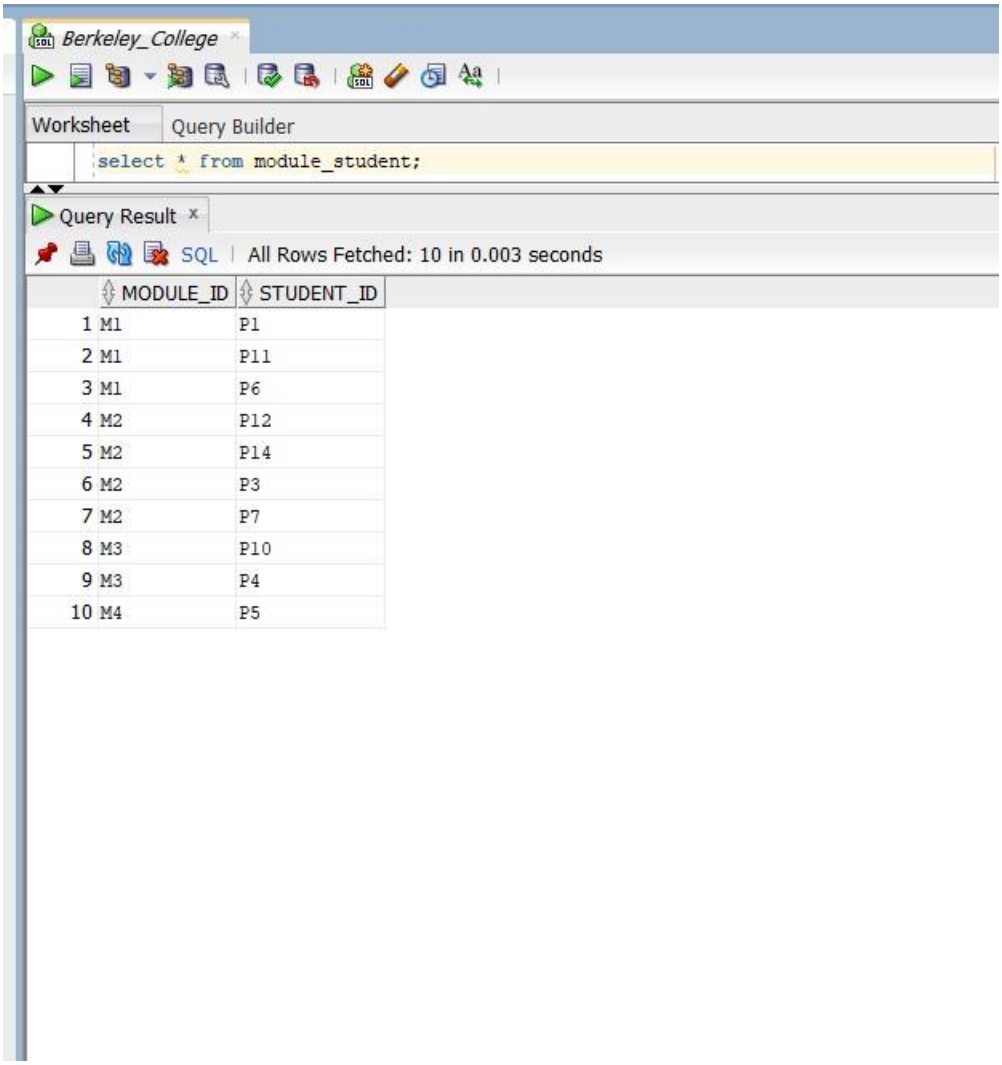
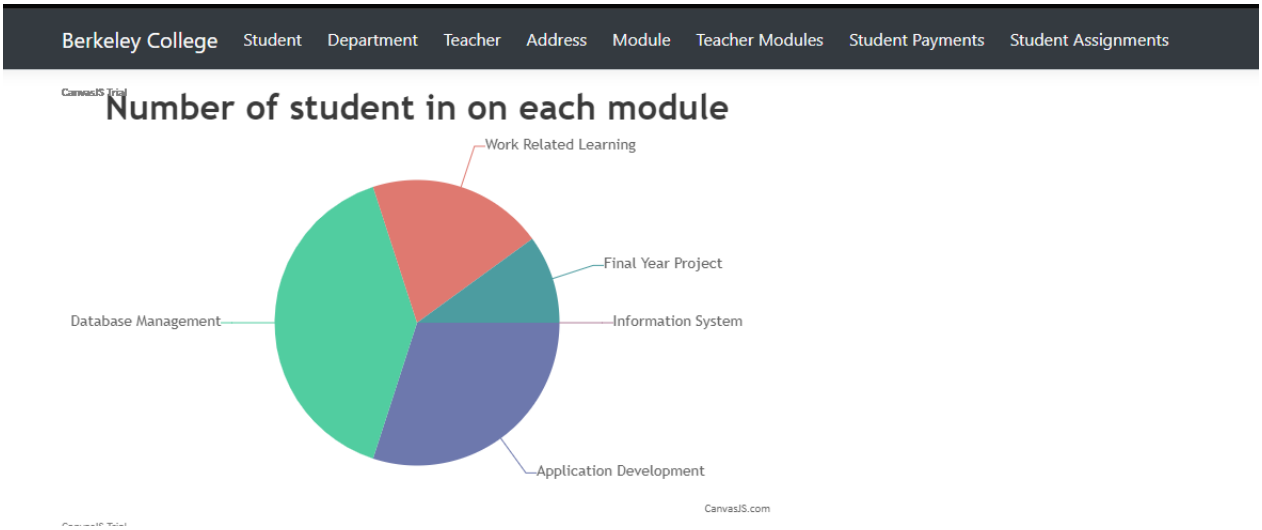
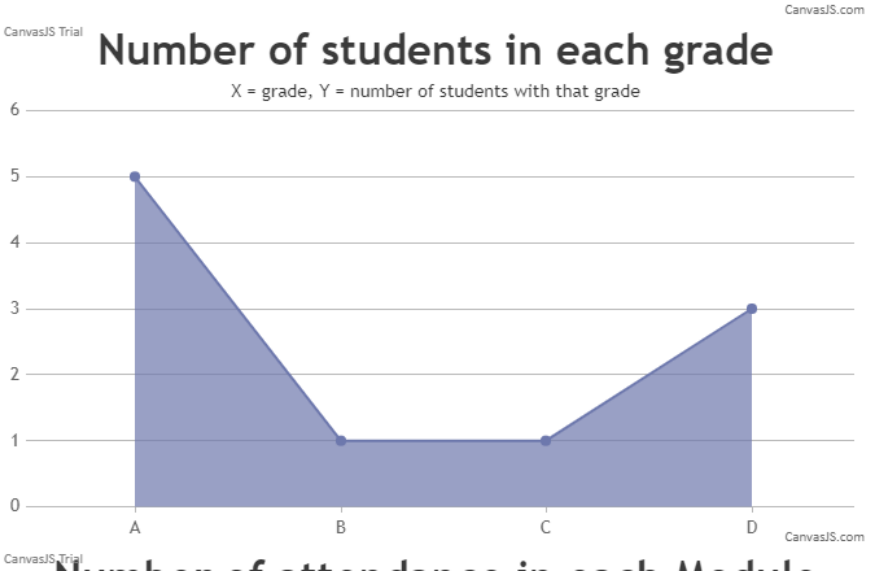


Figure 45: Select statement in Module_Student Table

Forms

Dashboard or Home Page





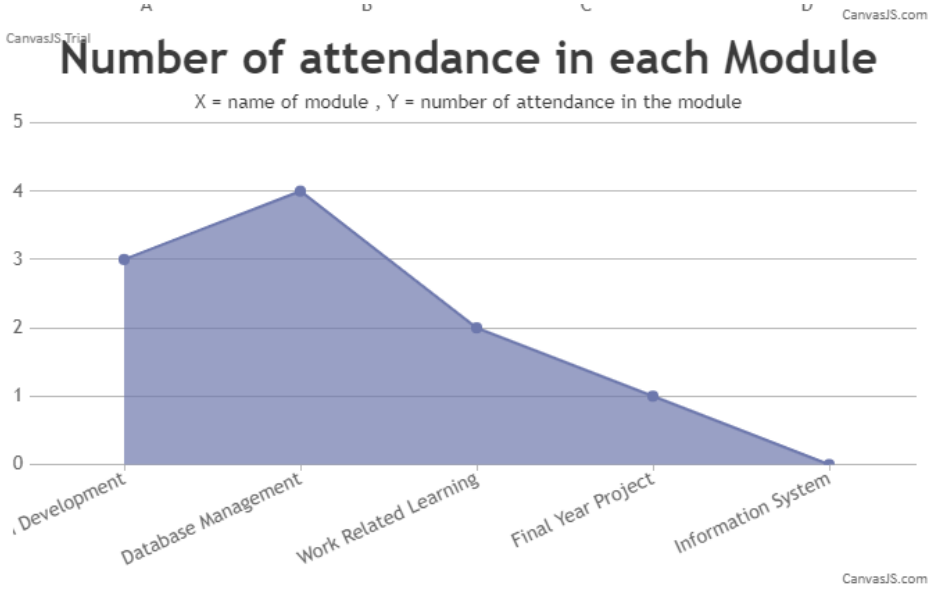


Figure 46: Dashboard Page of Berkeley College

Simple Forms

Simple Form of Student Table

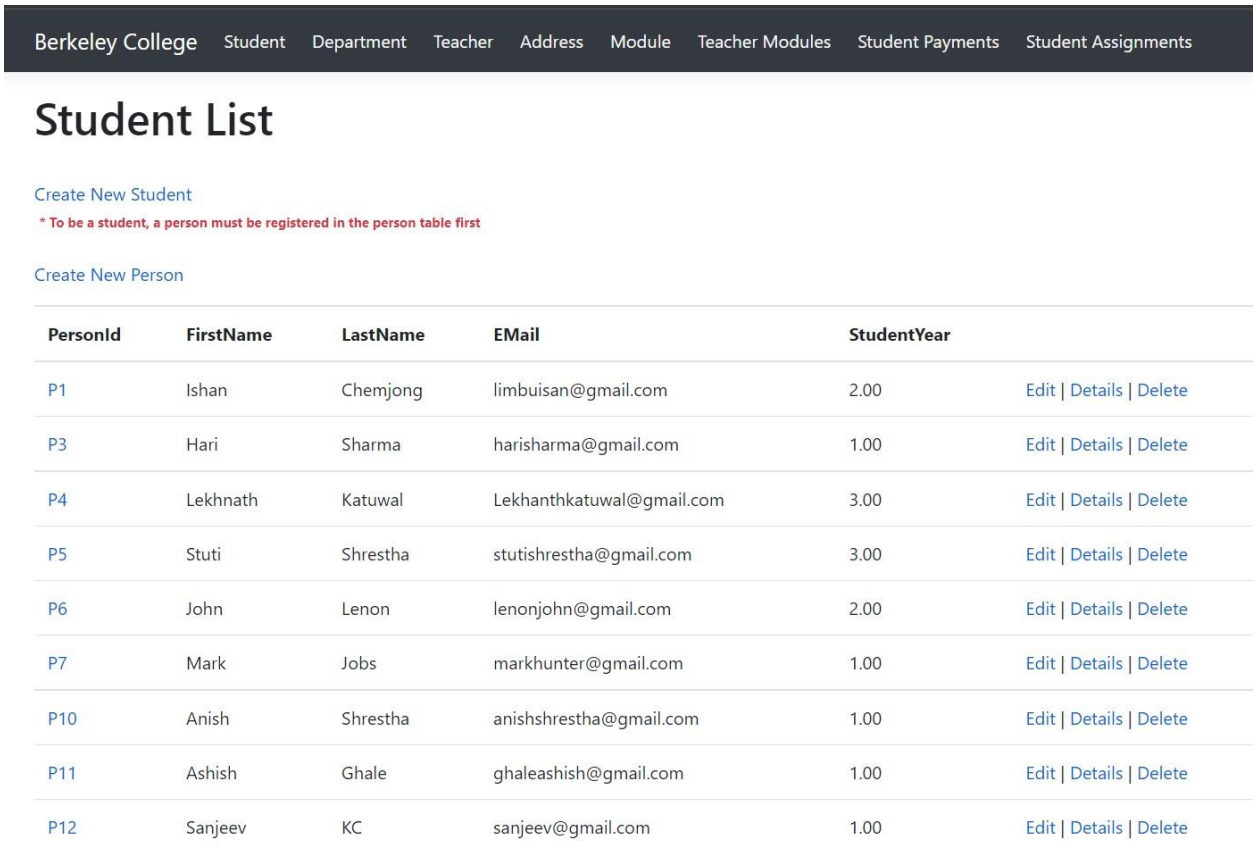


Figure 47: Simple Form of Student Table

Simple Form of Department Table

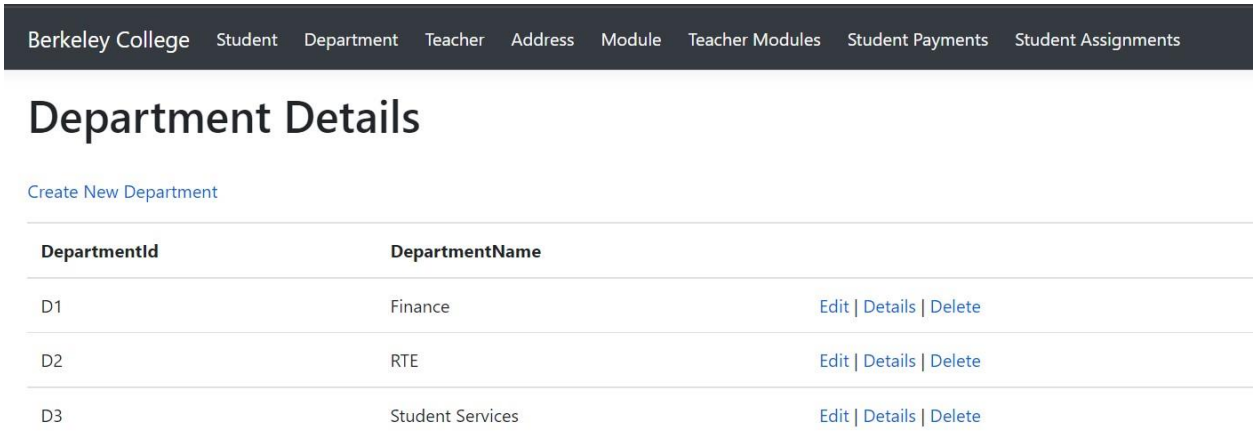


Figure 48: Simple Form of Department Table

Simple Form of Teacher Table

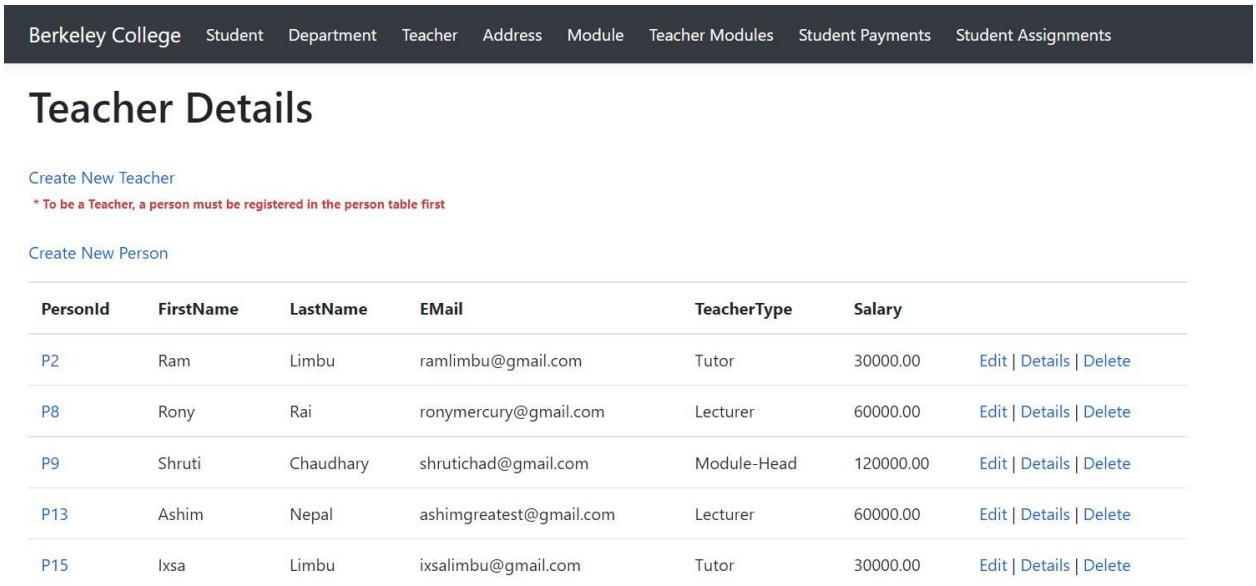


Figure 49: Simple Form of Teacher Table

Simple Form of Address Table

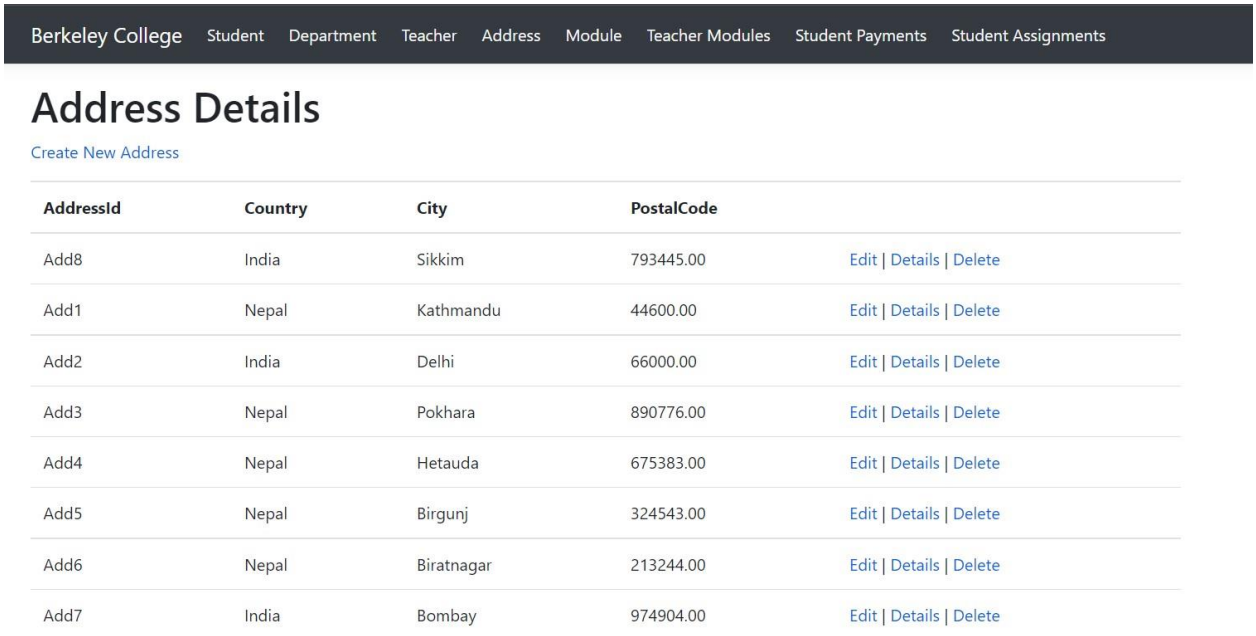


Figure 50: Simple Form of Address Table

Simple Form of Module Table

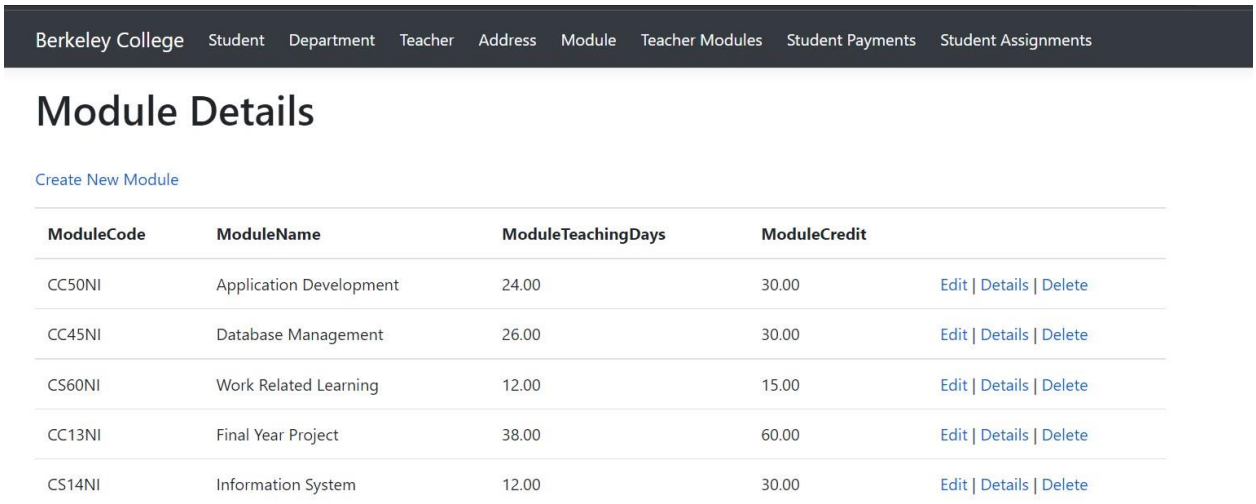


Figure 51: Simple Form of Module Table

Complex Forms and SQL Queries

Complex Forms

Complex Form for Teacher Module Mapping

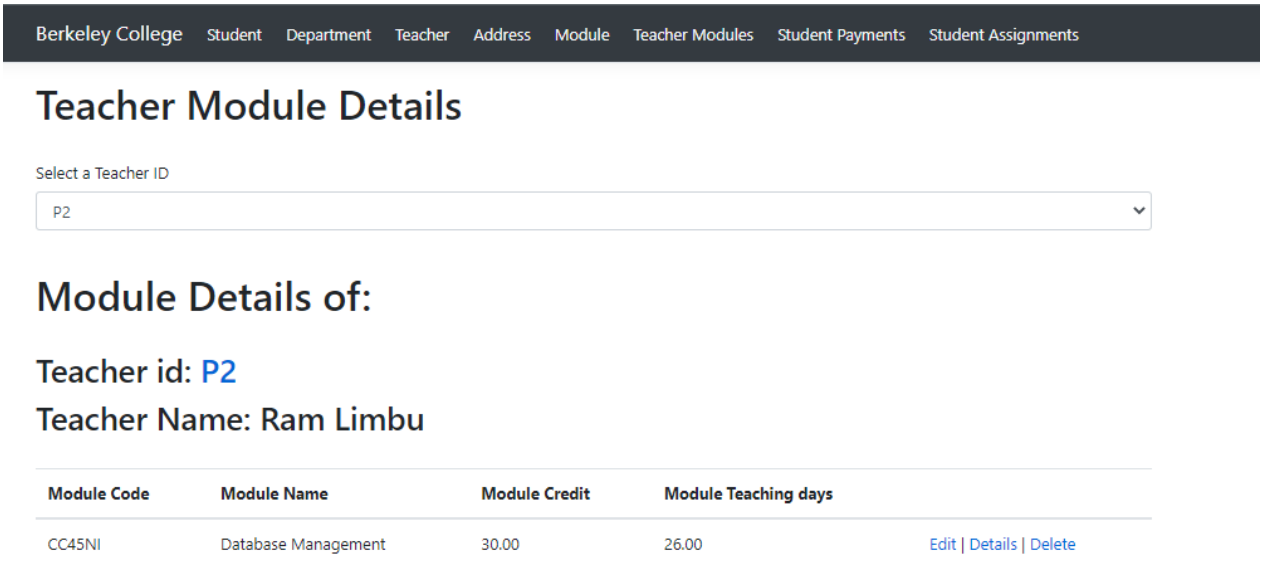


Figure 52: Complex Form for Teacher Module Mapping

Complex Form for Student Fee Payment Details

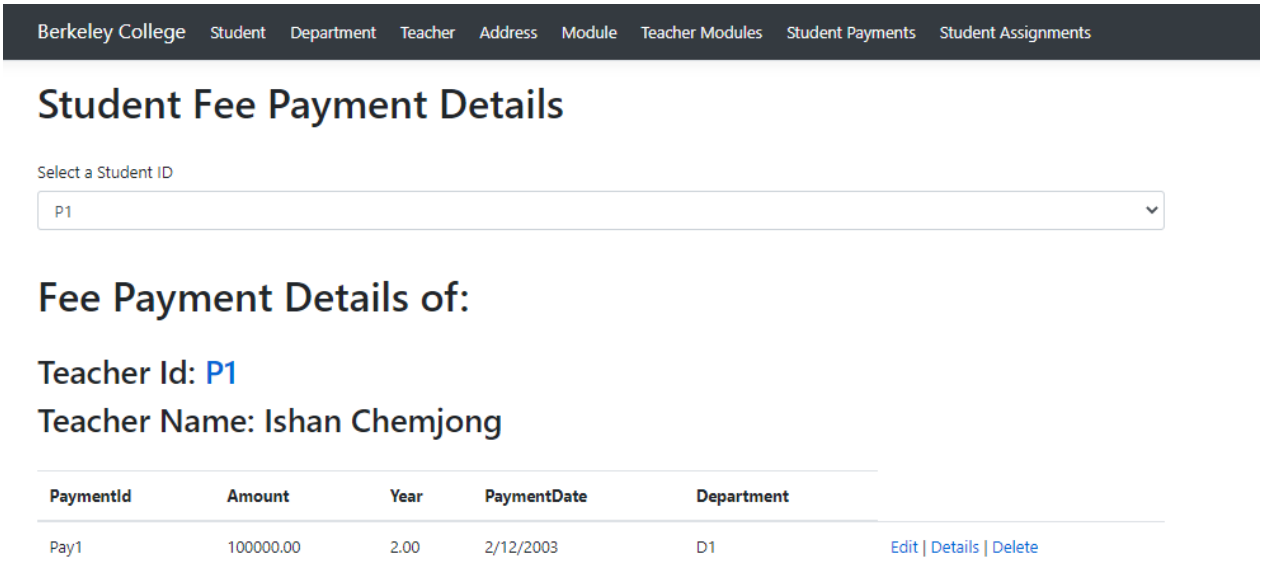


Figure 53: Complex Form for Student Fee Payment Details

Complex Form for Student Assignment Details

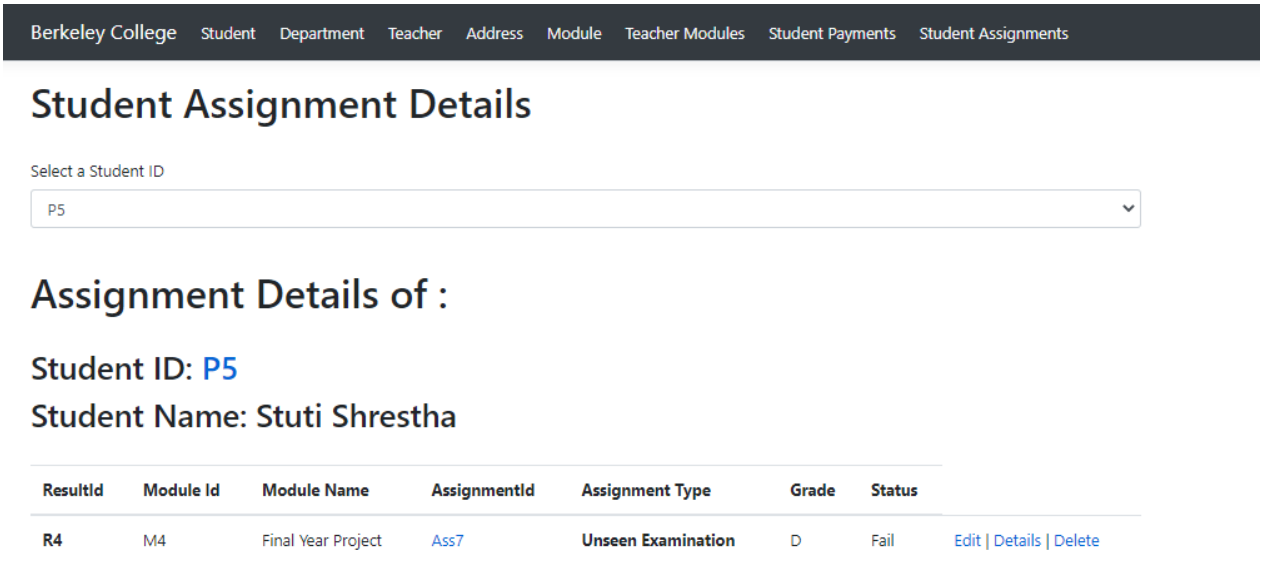


Figure 54: Complex Form for Student Assignment Details

SQL Queries

SQL Queries for Teacher Module Mapping

```
ViewData["TeacherId"] = new SelectList(_context.Teacher, "PersonId", "PersonId");

ViewBag.Teacher = (await _context.Teacher.FromSqlRaw($"select * from teacher where person_id= '{id}'").ToListAsync())[0]

ViewBag.Modules = await _context.Module.FromSqlRaw($"select * from module where module_id in (select module_id from teacher_module_info where teacher_id ='{id}')").ToListAsync();

ViewBag.Teacher.Person = (await _context.Person.FromSqlRaw($"select * from person where person_id = '{ViewBag.Teacher.PersonId}'").ToListAsync())[0];
```

SQL Queries for Student Fee Payment Details

```
ViewData["StudentId"] = new SelectList(_context.Student, "PersonId", "PersonId");

ViewBag.student = (await _context.Student.FromSqlRaw($"select * from student where person_id= '{id}'").ToListAsync())[0];

ViewBag.payments = await _context.StudentFeePayment.FromSqlRaw($"select * from student_fee_payment where student_id='{id}'").ToListAsync();

ViewBag.Student.Person = (await _context.Person.FromSqlRaw($"select * from person where person_id = '{ViewBag.Student.PersonId}'").ToListAsync())[0];
```

SQL Queries Form for Student Assignment Details

```
ViewData["StudentId"] = new SelectList(_context.Student, "PersonId", "PersonId");

ViewBag.student = (await _context.Student.FromSqlRaw($"select * from student where person_id='{id}'").ToListAsync())[0];

ViewBag.results = await _context.ResultInfo.FromSqlRaw($"select * from result_info where student_id='{id}'").ToListAsync();

ViewBag.Student.Person = (await _context.Person.FromSqlRaw($"select * from person where person_id ='{ViewBag.Student.PersonId}'").ToListAsync())[0];
```

As per the coursework, for every complex form I had to provide its SQL queries. However, as I have used MVC architecture for constructing the backend of my web application, showing only query will completely be irrelevant and make no sense. This is the reason why I have pasted the whole code where every query's output is converted into a list and stored in a new variable in the ViewBag. For writing the SQL query, I have used the FromSqlRaw function provided by Entity Framework Core as it is used to use and manipulate as compared to using LINQ.

Testing

Testing for Student Table

Reading data from Student Table

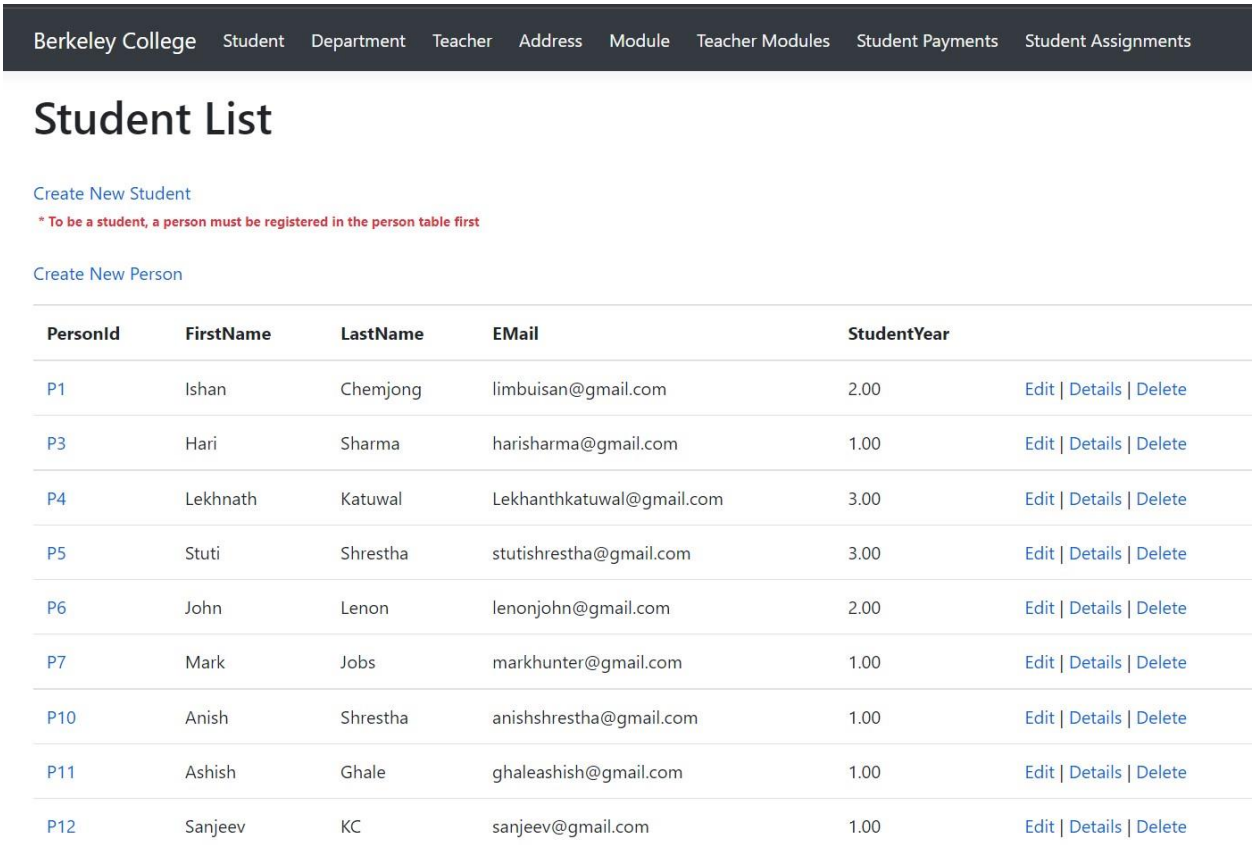


Figure 55: Reading data from Student Table

Create operation in Student Table

PersonId	FirstName	LastName	EMail	StudentYear	
P1	Ishan	Chemjong	limbuisan@gmail.com	2.00	Edit | Details | Delete
P3	Hari	Sharma	harisharma@gmail.com	1.00	Edit | Details | Delete
P4	Lekhnath	Katuwal	Lekhanthkatuwal@gmail.com	3.00	Edit | Details | Delete
P5	Stuti	Shrestha	stutishrestha@gmail.com	3.00	Edit | Details | Delete
P6	John	Lenon	lenonjohn@gmail.com	2.00	Edit | Details | Delete
P7	Mark	Jobs	markhunter@gmail.com	1.00	Edit | Details | Delete
P10	Anish	Shrestha	anishshrestha@gmail.com	1.00	Edit | Details | Delete
P11	Ashish	Ghale	ghaleashish@gmail.com	1.00	Edit | Details | Delete
P12	Sanjeev	KC	sanjeev@gmail.com	1.00	Edit | Details | Delete
P14	Anmol	Amatya	hondaanmol@gmail.com	2.00	Edit | Details | Delete
p21	Test	Tests	test@test.cpm	2.00	Edit | Details | Delete

Figure 56: Student Table

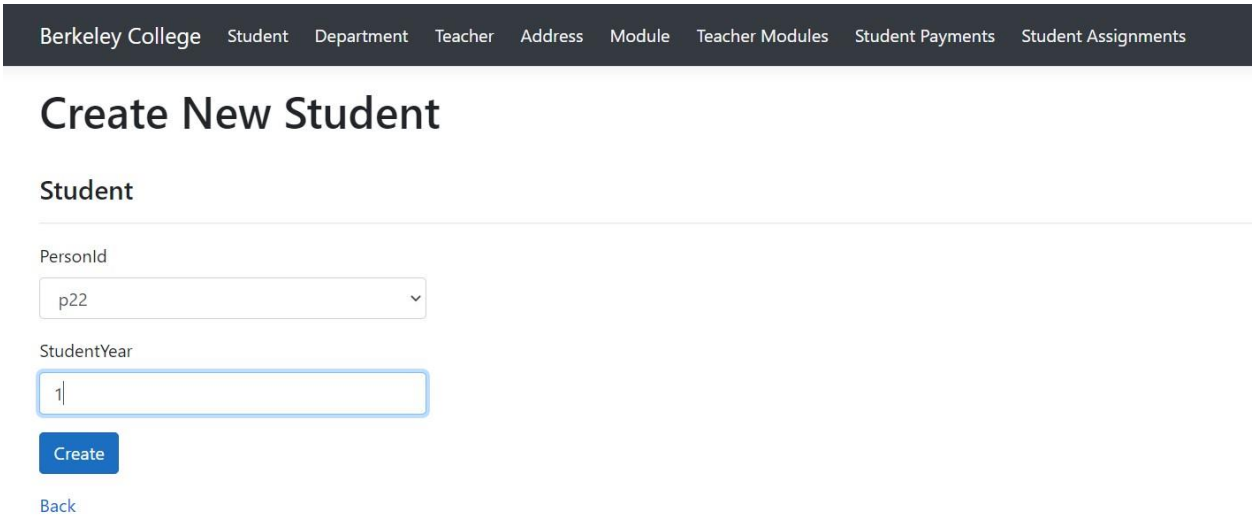


Figure 57: Inserting New Student Record

PersonId	FirstName	LastName	EMail	StudentYear	
P1	Ishan	Chemjong	limbuisan@gmail.com	2.00	Edit | Details | Delete
P3	Hari	Sharma	harisharma@gmail.com	1.00	Edit | Details | Delete
P4	Lekhnath	Katuwal	Lekhanthkatuwal@gmail.com	3.00	Edit | Details | Delete
P5	Stuti	Shrestha	stutishrestha@gmail.com	3.00	Edit | Details | Delete
P6	John	Lenon	lenonjohn@gmail.com	2.00	Edit | Details | Delete
P7	Mark	Jobs	markhunter@gmail.com	1.00	Edit | Details | Delete
P10	Anish	Shrestha	anishshrestha@gmail.com	1.00	Edit | Details | Delete
P11	Ashish	Ghale	ghaleashish@gmail.com	1.00	Edit | Details | Delete
P12	Sanjeev	KC	sanjeev@gmail.com	1.00	Edit | Details | Delete
P14	Anmol	Amatya	hondaanmol@gmail.com	2.00	Edit | Details | Delete
p22	Harry	Maguire	harryfoot@gmail.com	1.00	Edit | Details | Delete

Figure 58: Inserted New Student Record shown in table

Action	A new Student record is to be added in the Student table. Insert the records of the new Student and press Create button.
Expected Result	A new Student record is added in the table.
Actual Result	A new Student record was added in the table.
Conclusion	Test Successful

Table 14: Testing of create operation in Student Table

Update operation in Student Table

PersonId	FirstName	LastName	EMail	StudentYear	
P1	Ishan	Chemjong	limbuisan@gmail.com	2.00	Edit | Details | Delete
P3	Hari	Sharma	harisharma@gmail.com	1.00	Edit | Details | Delete
P4	Lekhnath	Katuwal	Lekhanthkatuwal@gmail.com	3.00	Edit | Details | Delete
P5	Stuti	Shrestha	stutishrestha@gmail.com	3.00	Edit | Details | Delete
P6	John	Lenon	lenonjohn@gmail.com	2.00	Edit | Details | Delete
P7	Mark	Jobs	markhunter@gmail.com	1.00	Edit | Details | Delete
P10	Anish	Shrestha	anishshrestha@gmail.com	1.00	Edit | Details | Delete
P11	Ashish	Ghale	ghaleashish@gmail.com	1.00	Edit | Details | Delete
P12	Sanjeev	KC	sanjeev@gmail.com	1.00	Edit | Details | Delete
P14	Anmol	Amatya	hondaanmol@gmail.com	2.00	Edit | Details | Delete
p22	Harry	Maguire	harryfoot@gmail.com	1.00	Edit | Details | Delete

Figure 59: Student Table

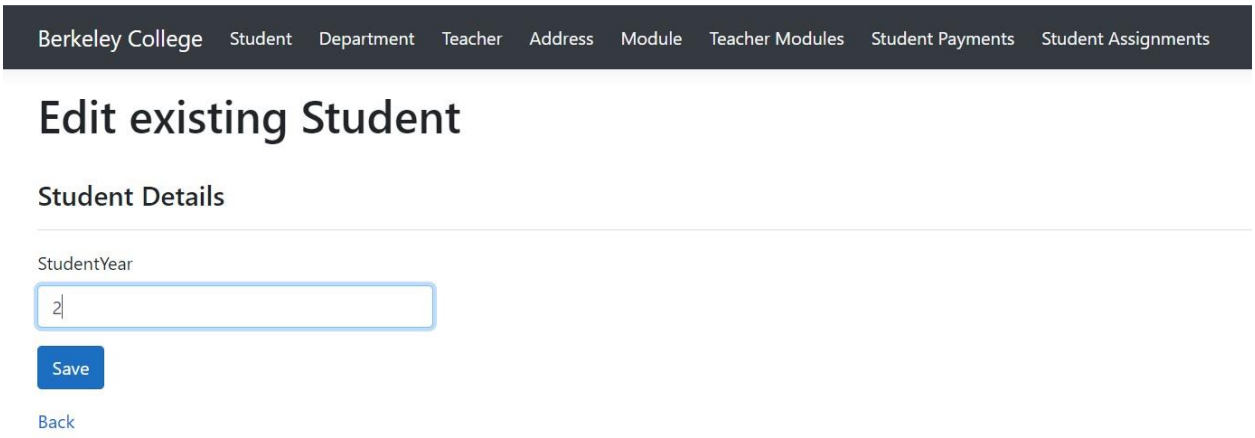


Figure 60: Updating a Student Record

PersonId	FirstName	LastName	EMail	StudentYear	
P1	Ishan	Chemjong	limbuisan@gmail.com	2.00	Edit | Details | Delete
P3	Hari	Sharma	harisharma@gmail.com	1.00	Edit | Details | Delete
P4	Lekhnath	Katuwal	Lekhanthkatuwal@gmail.com	3.00	Edit | Details | Delete
P5	Stuti	Shrestha	stutishrestha@gmail.com	3.00	Edit | Details | Delete
P6	John	Lenon	lenonjohn@gmail.com	2.00	Edit | Details | Delete
P7	Mark	Jobs	markhunter@gmail.com	1.00	Edit | Details | Delete
P10	Anish	Shrestha	anishshrestha@gmail.com	1.00	Edit | Details | Delete
P11	Ashish	Ghale	ghaleashish@gmail.com	1.00	Edit | Details | Delete
P12	Sanjeev	KC	sanjeev@gmail.com	1.00	Edit | Details | Delete
P14	Anmol	Amatya	hondaanmol@gmail.com	2.00	Edit | Details | Delete
p22	Harry	Maguire	harryfoot@gmail.com	2.00	Edit | Details | Delete

Figure 61: Updated Student Record shown in table

Action	A Student record is to be updated in the Student table. Insert the new records of the Student and press Edit button.
Expected Result	The Student record is updated in the table.
Actual Result	The Student record was updated in the table.
Conclusion	Test Successful

Table 15: Testing of update operation in Student Table

Delete operation in Student Table

PersonId	FirstName	LastName	EMail	StudentYear	
P1	Ishan	Chemjong	limbuisan@gmail.com	2.00	Edit | Details | Delete
P3	Hari	Sharma	harisharma@gmail.com	1.00	Edit | Details | Delete
P4	Lekhnath	Katuwal	Lekhanthkatuwal@gmail.com	3.00	Edit | Details | Delete
P5	Stuti	Shrestha	stutishrestha@gmail.com	3.00	Edit | Details | Delete
P6	John	Lenon	lenonjohn@gmail.com	2.00	Edit | Details | Delete
P7	Mark	Jobs	markhunter@gmail.com	1.00	Edit | Details | Delete
P10	Anish	Shrestha	anishshrestha@gmail.com	1.00	Edit | Details | Delete
P11	Ashish	Ghale	ghaleashish@gmail.com	1.00	Edit | Details | Delete
P12	Sanjeev	KC	sanjeev@gmail.com	1.00	Edit | Details | Delete
P14	Anmol	Amatya	hondaanmol@gmail.com	2.00	Edit | Details | Delete
p22	Harry	Maguire	harryfoot@gmail.com	2.00	Edit | Details | Delete

Figure 62: Student Table before deleting a Student Record

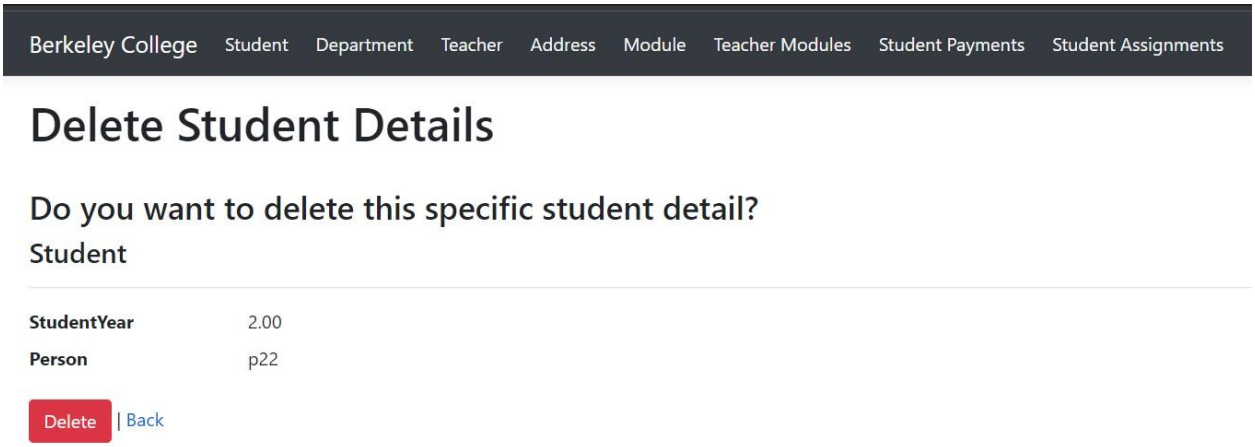


Figure 63: Deleting a Student Record

PersonId	FirstName	LastName	EMail	StudentYear	
P1	Ishan	Chemjong	limbuisan@gmail.com	2.00	Edit | Details | Delete
P3	Hari	Sharma	harisharma@gmail.com	1.00	Edit | Details | Delete
P4	Lekhnath	Katuwal	Lekhanthkatuwal@gmail.com	3.00	Edit | Details | Delete
P5	Stuti	Shrestha	stutishrestha@gmail.com	3.00	Edit | Details | Delete
P6	John	Lenon	lenonjohn@gmail.com	2.00	Edit | Details | Delete
P7	Mark	Jobs	markhunter@gmail.com	1.00	Edit | Details | Delete
P10	Anish	Shrestha	anishshrestha@gmail.com	1.00	Edit | Details | Delete
P11	Ashish	Ghale	ghaleashish@gmail.com	1.00	Edit | Details | Delete
P12	Sanjeev	KC	sanjeev@gmail.com	1.00	Edit | Details | Delete
P14	Anmol	Amatya	hondaanmol@gmail.com	2.00	Edit | Details | Delete

Figure 64: Student Table after deleting a Student Record

Action	A Student record is to be deleted in the Student table. Press the Delete button in the Student record's row.
Expected Result	The Student record is deleted from the table.
Actual Result	The Student record was deleted from the table.
Conclusion	Test Successful

Table 16: Testing of delete operation in Student Table

Testing for Department Table

Reading data from Department Table

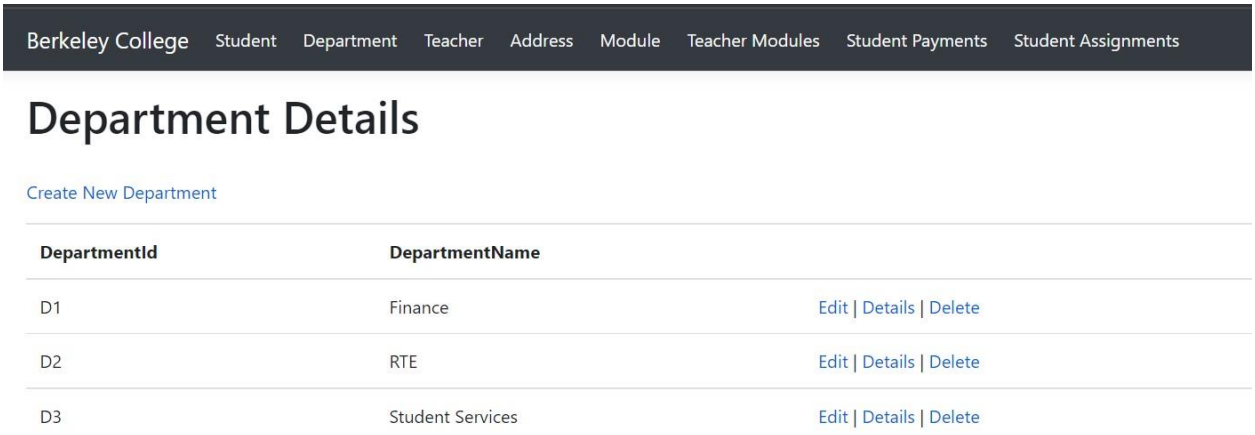


Figure 65: Reading data from Department Table

Create operation in Department Table

DepartmentId	DepartmentName	
D1	Finance	Edit | Details | Delete
D2	RTE	Edit | Details | Delete
D3	Student Services	Edit | Details | Delete

Figure 66: Department Table

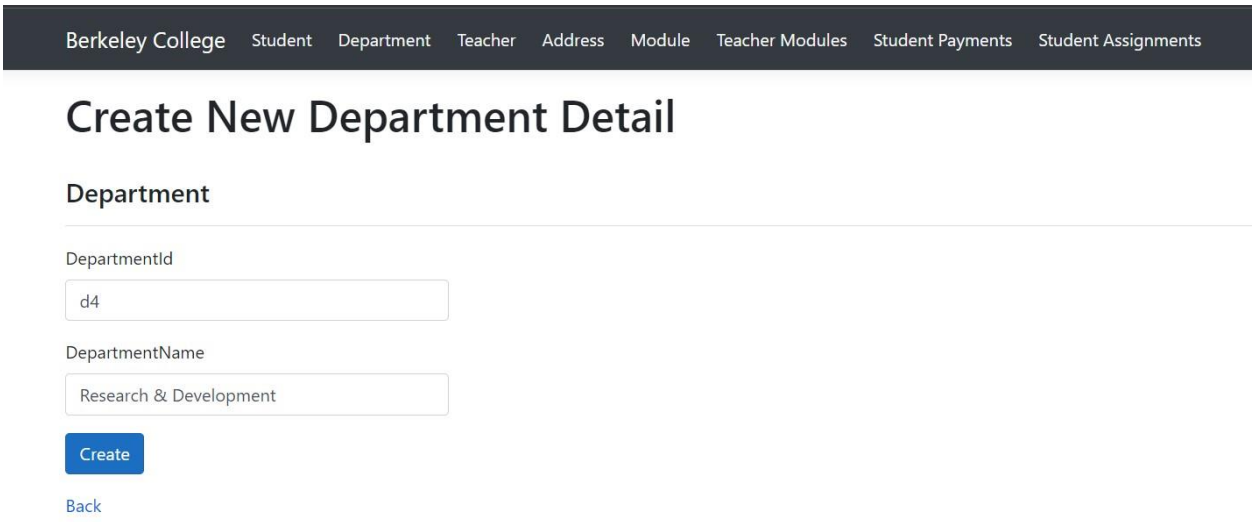


Figure 67: Inserting New Department Record

DepartmentId	DepartmentName	
D1	Finance	Edit | Details | Delete
D2	RTE	Edit | Details | Delete
D3	Student Services	Edit | Details | Delete
D4	Research & Development	Edit | Details | Delete

Figure 68: Inserting New Department Record shown in table

Action	A new Department record is to be added in the Department table. Insert the records of the new Department and press Create button.
Expected Result	A new Department record is added in the table.
Actual Result	A new Department record was added in the table.
Conclusion	Test Successful

Table 17: Testing of create operation in Department Table

Update operation in Department Table

DepartmentId	DepartmentName	
D1	Finance	Edit | Details | Delete
D2	RTE	Edit | Details | Delete
D3	Student Services	Edit | Details | Delete
D4	Research & Development	Edit | Details | Delete

Figure 69: Department Table

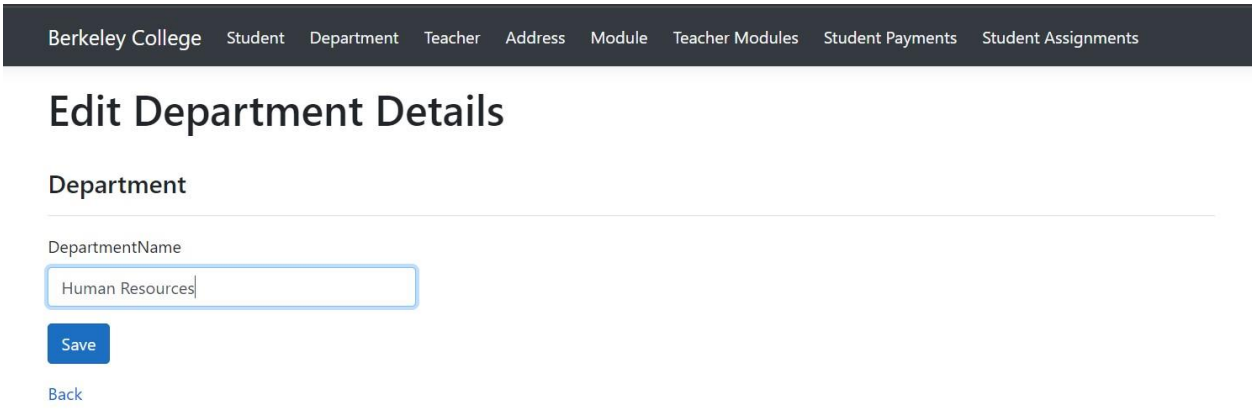


Figure 70: Updating a Department Record

DepartmentId	DepartmentName	
D1	Finance	Edit | Details | Delete
D2	Human Resources	Edit | Details | Delete
D3	Student Services	Edit | Details | Delete
D4	Research & Development	Edit | Details | Delete

Figure 71: Updated Department Record shown in table

Action	A Department record is to be updated in the Department table. Insert the new records of the Department and press Edit button.
Expected Result	The Department record is updated in the table.
Actual Result	The Department record was updated in the table.
Conclusion	Test Successful

Table 18: Testing of update operation in Department Table

Delete operation in Department Table

DepartmentId	DepartmentName	
D1	Finance	Edit | Details | Delete
D2	Human Resources	Edit | Details | Delete
D3	Student Services	Edit | Details | Delete
D4	Research & Development	Edit | Details | Delete

Figure 72: Department Table before deleting a Department Record

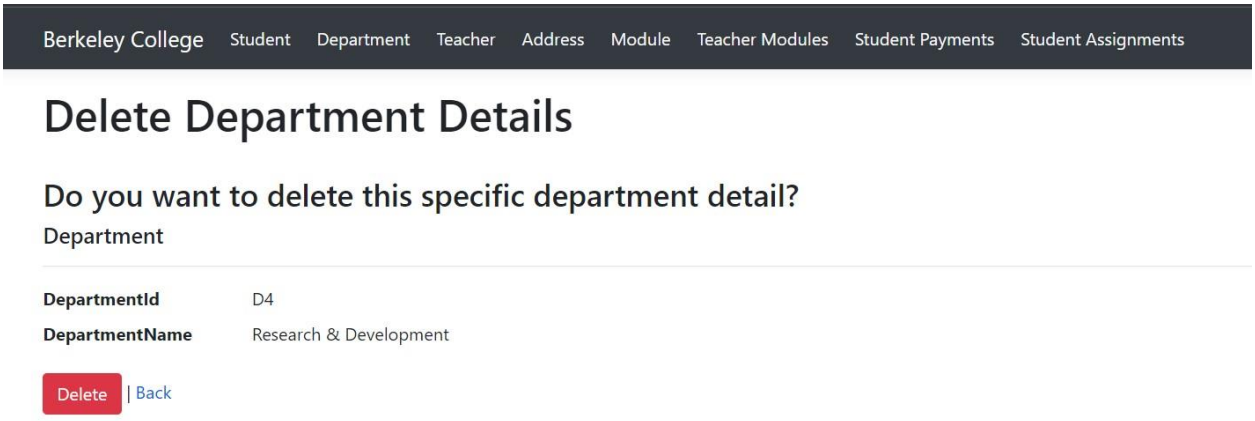


Figure 73: Deleting a Department Record

DepartmentId	DepartmentName	
D1	Finance	Edit | Details | Delete
D2	Human Resources	Edit | Details | Delete
D3	Student Services	Edit | Details | Delete

Figure 74: Department Table after deleting a Department Record

Action	A Department record is to be deleted in the Department table. Press the Delete button in the Department record's row.
Expected Result	The Department record is deleted from the table.
Actual Result	The Department record was deleted from the table.
Conclusion	Test Successful

Table 19 : Testing of delete operation in Department Table

Testing for Teacher Table

Reading data from Teacher Table

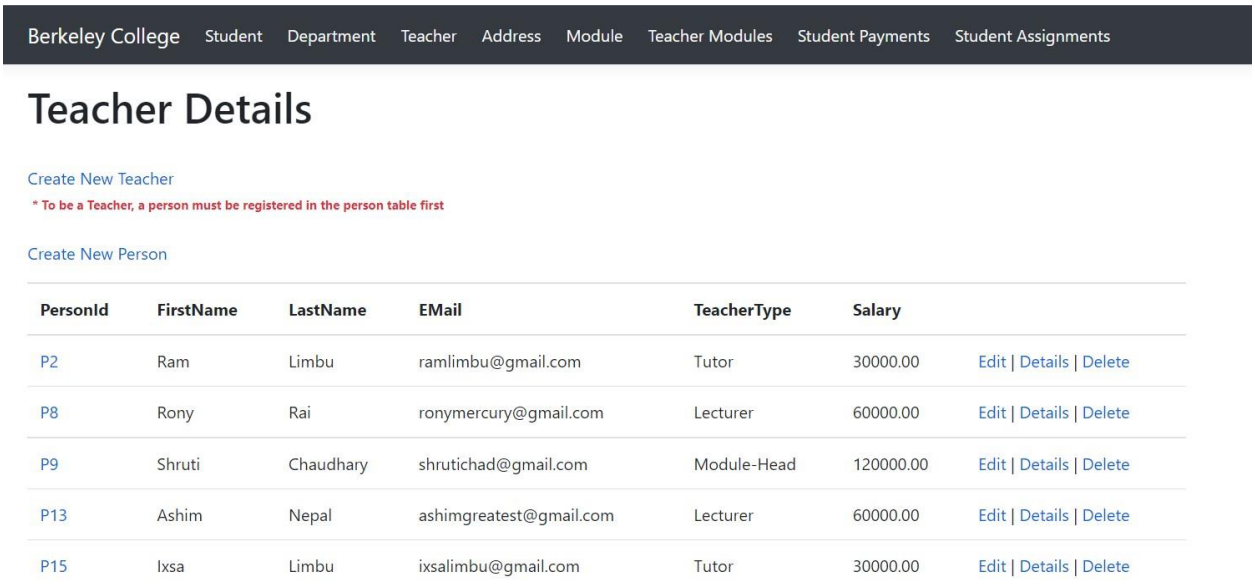


Figure 75: Reading data from Teacher Table

Create operation in Teacher Table

PersonId	FirstName	LastName	EMail	TeacherType	Salary	
P2	Ram	Limbu	ramlimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
P8	Rony	Rai	ronymercury@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P9	Shruti	Chaudhary	shrutichad@gmail.com	Module-Head	120000.00	Edit | Details | Delete
P13	Ashim	Nepal	ashimgreatest@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P15	Ixsa	Limbu	ixsalimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete

Figure 76: Teacher Table

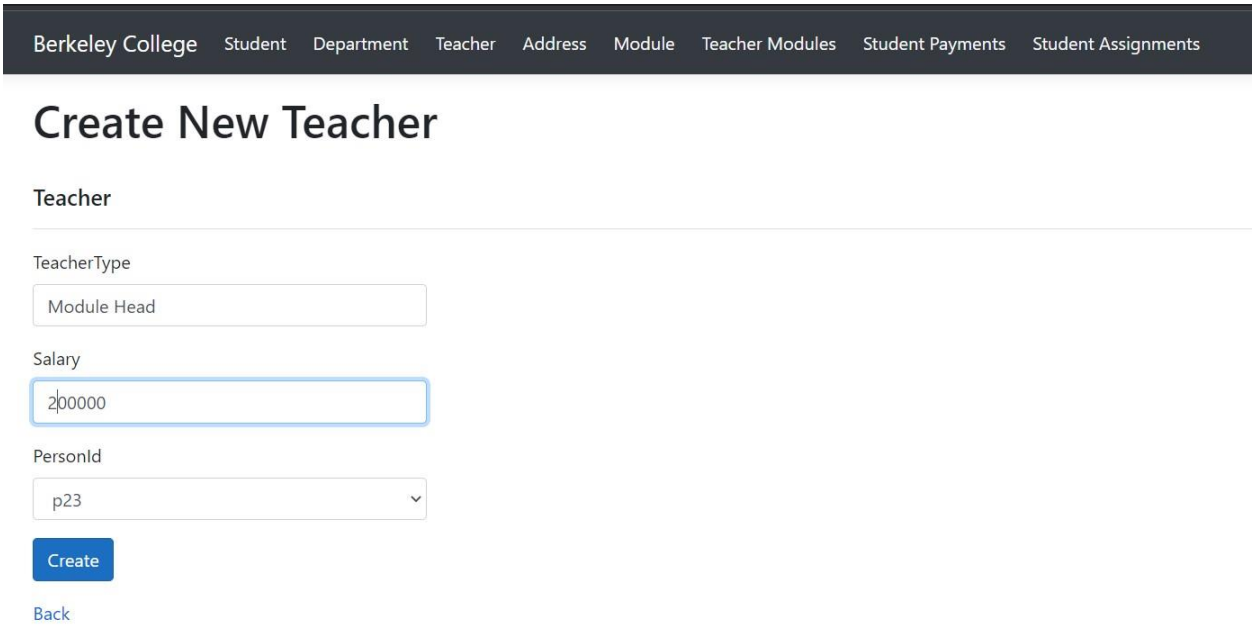


Figure 77: Inserting New Teacher Record

PersonId	FirstName	LastName	EMail	TeacherType	Salary	
P2	Ram	Limbu	ramlimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
P8	Rony	Rai	ronymercury@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P9	Shruti	Chaudhary	shrutichad@gmail.com	Module-Head	120000.00	Edit | Details | Delete
P13	Ashim	Nepal	ashimgreatest@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P15	Ixsa	Limbu	ixsalimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
p23	Sulav	Budhathoki	sulavceo@gmail.com	Module Head	200000.00	Edit | Details | Delete

Figure 78: Inserting New Teacher Record shown in table

Action	A new Teacher record is to be added in the Teacher table. Insert the records of the new Teacher and press Create button.
Expected Result	A new Teacher record is added in the table.
Actual Result	A new Teacher record was added in the table.
Conclusion	Test Successful

Table 20: Testing of create operation in Teacher Table

Update operation in Teacher Table

PersonId	FirstName	LastName	EMail	TeacherType	Salary	
P2	Ram	Limbu	ramlimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
P8	Rony	Rai	ronymercury@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P9	Shruti	Chaudhary	shrutichad@gmail.com	Module-Head	120000.00	Edit | Details | Delete
P13	Ashim	Nepal	ashimgreatest@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P15	Ixsa	Limbu	ixsalimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
p23	Sulav	Budhathoki	sulavceo@gmail.com	Module Head	200000.00	Edit | Details | Delete

Figure 79: Teacher Table

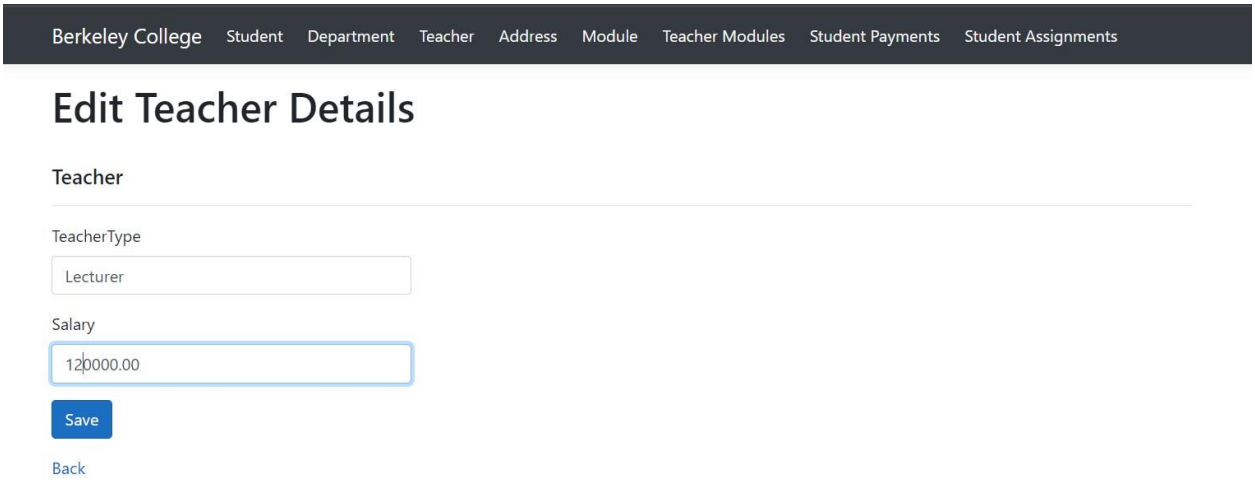


Figure 80: Updating a Teacher Record

PersonId	FirstName	LastName	EMail	TeacherType	Salary	
P2	Ram	Limbu	ramlimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
P8	Rony	Rai	ronymercury@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P9	Shruti	Chaudhary	shrutichad@gmail.com	Module-Head	120000.00	Edit | Details | Delete
P13	Ashim	Nepal	ashimgreatest@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P15	Ixsa	Limbu	ixsalimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
p23	Sulav	Budhathoki	sulavceo@gmail.com	Lecturer	120000.00	Edit | Details | Delete

Figure 81: Updated Teacher Record shown in table

Action	A Teacher record is to be updated in the Teacher table. Insert the new records of the Teacher and press Edit button.
Expected Result	The Teacher record is updated in the table.
Actual Result	The Teacher record was updated in the table.
Conclusion	Test Successful

Table 21: Testing of update operation in Teacher Table

Delete operation in Teacher Table

PersonId	FirstName	LastName	EMail	TeacherType	Salary	
P2	Ram	Limbu	ramlimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
P8	Rony	Rai	ronymercury@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P9	Shruti	Chaudhary	shrutichad@gmail.com	Module-Head	120000.00	Edit | Details | Delete
P13	Ashim	Nepal	ashimgreatest@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P15	Ixsa	Limbu	ixsalimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
p23	Sulav	Budhathoki	sulavceo@gmail.com	Lecturer	120000.00	Edit | Details | Delete

Figure 82: Teacher Table before deleting a Teacher Record

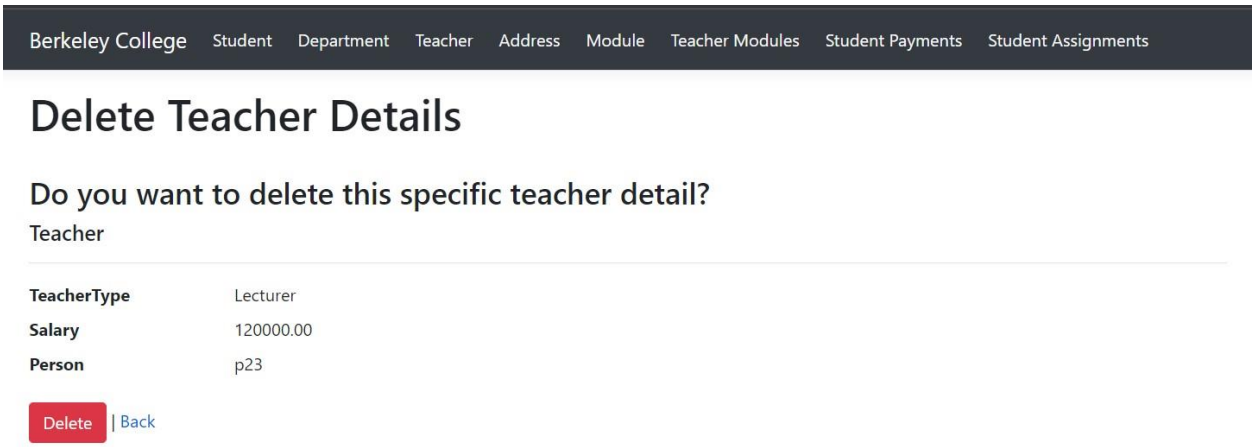


Figure 83: Deleting a Teacher Record

PersonId	FirstName	LastName	EMail	TeacherType	Salary	
P2	Ram	Limbu	ramlimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete
P8	Rony	Rai	ronymercury@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P9	Shruti	Chaudhary	shrutichad@gmail.com	Module-Head	120000.00	Edit | Details | Delete
P13	Ashim	Nepal	ashimgreatest@gmail.com	Lecturer	60000.00	Edit | Details | Delete
P15	Ixsa	Limbu	ixsalimbu@gmail.com	Tutor	30000.00	Edit | Details | Delete

Figure 84: Teacher Table after deleting a Teacher Record

Action	A Teacher record is to be deleted in the Teacher table. Press the Delete button in the Teacher record's row.
Expected Result	The Teacher record is deleted from the table.
Actual Result	The Teacher record was deleted from the table.
Conclusion	Test Successful

Table 22: Testing of delete operation in Teacher Table

Testing for Address Table

Reading data from Address Table

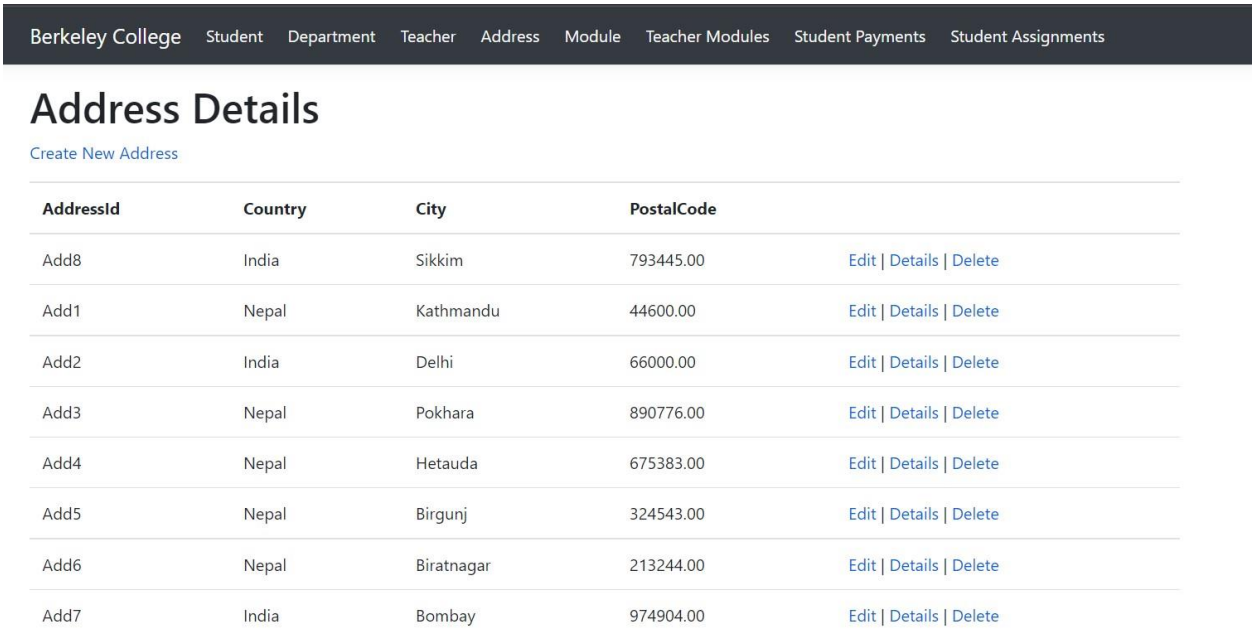


Figure 85: Reading data from Address Table

Create operation in Address Table

AddressId	Country	City	PostalCode	
Add8	India	Sikkim	793445.00	Edit | Details | Delete
Add1	Nepal	Kathmandu	44600.00	Edit | Details | Delete
Add2	India	Delhi	66000.00	Edit | Details | Delete
Add3	Nepal	Pokhara	890776.00	Edit | Details | Delete
Add4	Nepal	Hetauda	675383.00	Edit | Details | Delete
Add5	Nepal	Birgunj	324543.00	Edit | Details | Delete
Add6	Nepal	Biratnagar	213244.00	Edit | Details | Delete
Add7	India	Bombay	974904.00	Edit | Details | Delete

Figure 86: Address Table

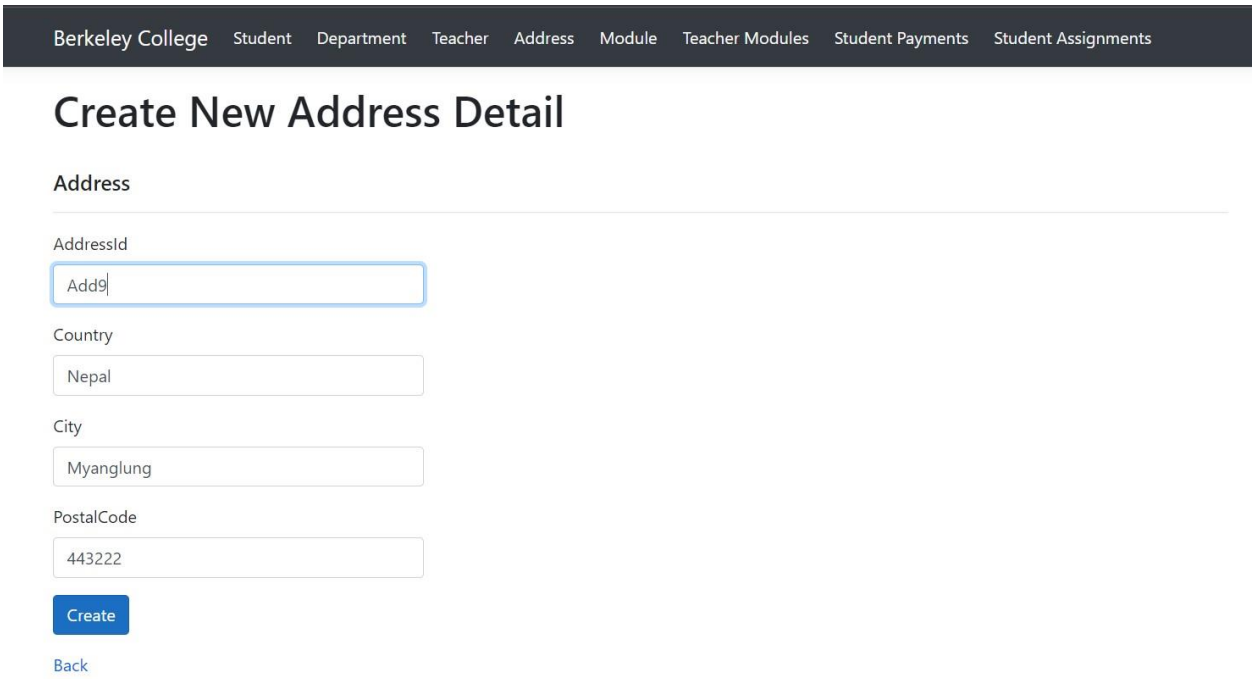


Figure 87: Inserting New Address Record

AddressId	Country	City	PostalCode	
Add8	India	Sikkim	793445.00	Edit | Details | Delete
Add1	Nepal	Kathmandu	44600.00	Edit | Details | Delete
Add2	India	Delhi	66000.00	Edit | Details | Delete
Add3	Nepal	Pokhara	890776.00	Edit | Details | Delete
Add4	Nepal	Hetauda	675383.00	Edit | Details | Delete
Add5	Nepal	Birgunj	324543.00	Edit | Details | Delete
Add6	Nepal	Biratnagar	213244.00	Edit | Details | Delete
Add7	India	Bombay	974904.00	Edit | Details | Delete
Add9	Nepal	Myanglung	443222.00	Edit | Details | Delete

Figure 88: Inserting New Address Record shown in table

Action	A new Address record is to be added in the Address table. Insert the records of the new Address and press Create button.
Expected Result	A new Address record is added in the table.
Actual Result	A new Address record was added in the table.
Conclusion	Test Successful

Table 23: Testing of create operation in Address Table

Update operation in Address Table

AddressId	Country	City	PostalCode	
Add8	India	Sikkim	793445.00	Edit | Details | Delete
Add1	Nepal	Kathmandu	44600.00	Edit | Details | Delete
Add2	India	Delhi	66000.00	Edit | Details | Delete
Add3	Nepal	Pokhara	890776.00	Edit | Details | Delete
Add4	Nepal	Hetauda	675383.00	Edit | Details | Delete
Add5	Nepal	Birgunj	324543.00	Edit | Details | Delete
Add6	Nepal	Biratnagar	213244.00	Edit | Details | Delete
Add7	India	Bombay	974904.00	Edit | Details | Delete
Add9	Nepal	Myanglung	443222.00	Edit | Details | Delete

Figure 89: Address Table

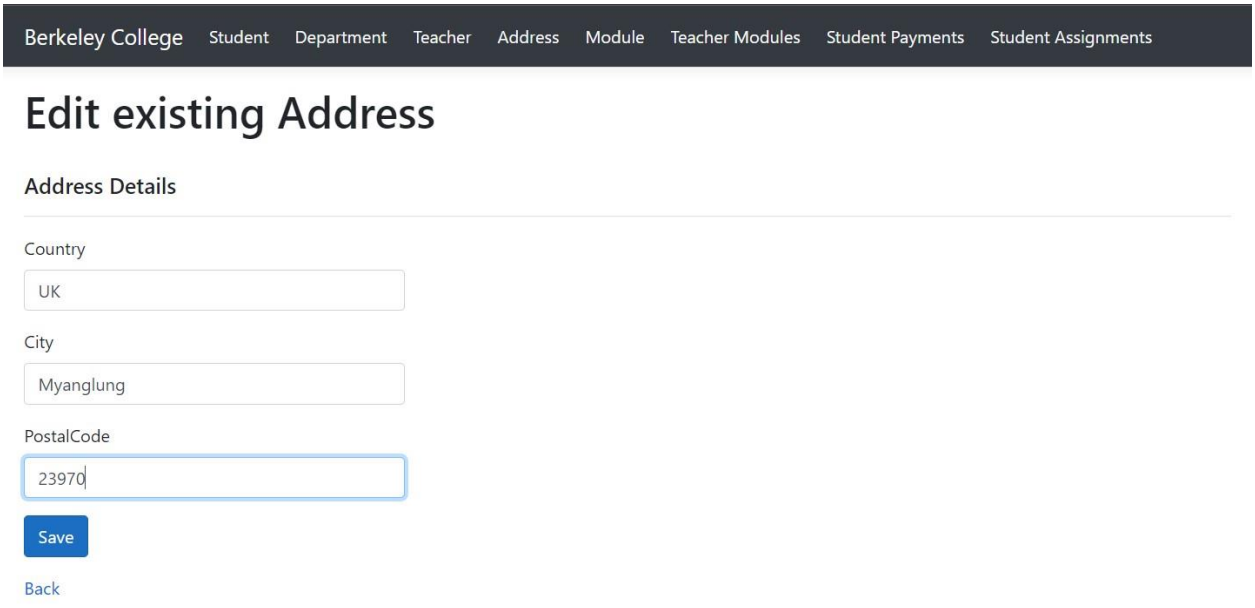


Figure 90: Updating an Address Record

AddressId	Country	City	PostalCode	
Add8	India	Sikkim	793445.00	Edit | Details | Delete
Add1	Nepal	Kathmandu	44600.00	Edit | Details | Delete
Add2	India	Delhi	66000.00	Edit | Details | Delete
Add3	Nepal	Pokhara	890776.00	Edit | Details | Delete
Add4	Nepal	Hetauda	675383.00	Edit | Details | Delete
Add5	Nepal	Birgunj	324543.00	Edit | Details | Delete
Add6	Nepal	Biratnagar	213244.00	Edit | Details | Delete
Add7	India	Bombay	974904.00	Edit | Details | Delete
Add9	UK	Myanglung	23970.00	Edit | Details | Delete

Figure 91: Updated Address Record shown in table

Action	An Address record is to be updated in the Address table. Insert the new records of the Address and press Edit button.
Expected Result	The Address record is updated in the table.
Actual Result	The Address record was updated in the table.
Conclusion	Test Successful

Figure 92: Testing of update operation in Address Table

Delete operation in Address Table

AddressId	Country	City	PostalCode	
Add8	India	Sikkim	793445.00	Edit | Details | Delete
Add1	Nepal	Kathmandu	44600.00	Edit | Details | Delete
Add2	India	Delhi	66000.00	Edit | Details | Delete
Add3	Nepal	Pokhara	890776.00	Edit | Details | Delete
Add4	Nepal	Hetauda	675383.00	Edit | Details | Delete
Add5	Nepal	Birgunj	324543.00	Edit | Details | Delete
Add6	Nepal	Biratnagar	213244.00	Edit | Details | Delete
Add7	India	Bombay	974904.00	Edit | Details | Delete
Add9	UK	Myanglung	23970.00	Edit | Details | Delete

Figure 93: Address Table before deleting an Address Record

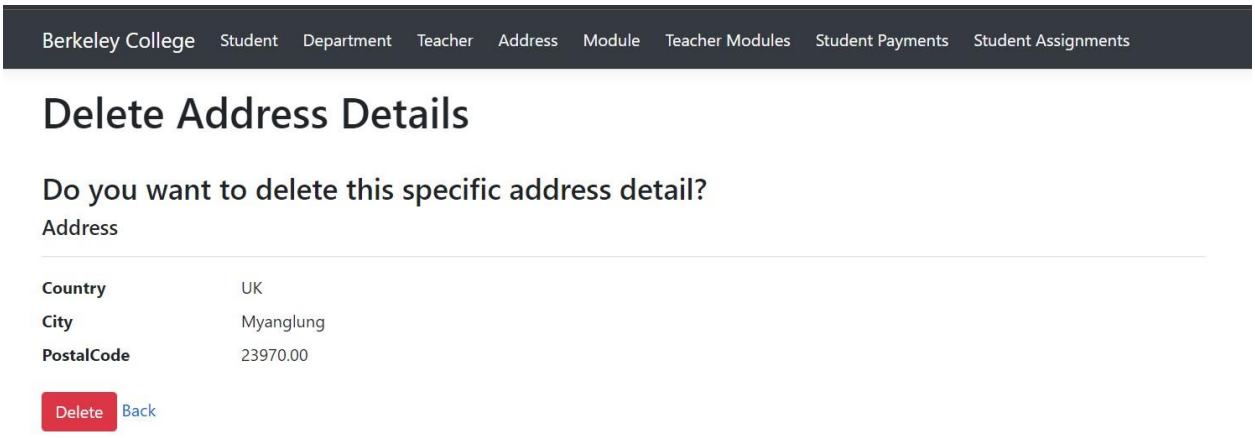


Figure 94: Deleting an Address Record

AddressId	Country	City	PostalCode	
Add8	India	Sikkim	793445.00	Edit | Details | Delete
Add1	Nepal	Kathmandu	44600.00	Edit | Details | Delete
Add2	India	Delhi	66000.00	Edit | Details | Delete
Add3	Nepal	Pokhara	890776.00	Edit | Details | Delete
Add4	Nepal	Hetauda	675383.00	Edit | Details | Delete
Add5	Nepal	Birgunj	324543.00	Edit | Details | Delete
Add6	Nepal	Biratnagar	213244.00	Edit | Details | Delete
Add7	India	Bombay	974904.00	Edit | Details | Delete

Figure 95: Address Table after deleting an Address Record

Action	An Address record is to be deleted in the Address table. Press the Delete button in the Address record's row.
Expected Result	The Address record is deleted from the table.
Actual Result	The Address record was deleted from the table.
Conclusion	Test Successful

Table 24: Testing of delete operation in Address Table

Testing for Module Table

Reading data from Module Table

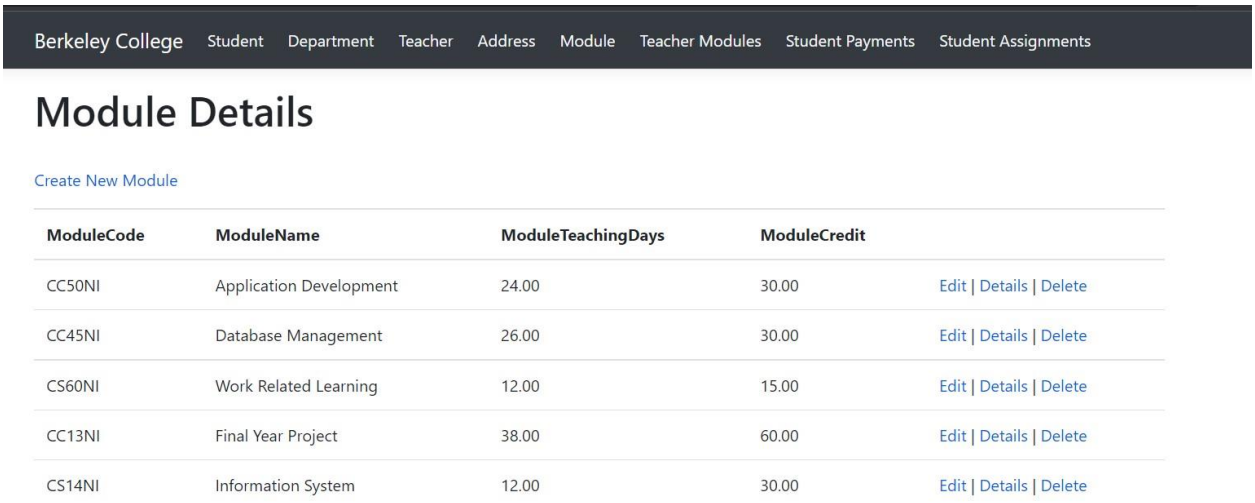


Figure 96: Reading data from Module Table

Create operation in Module Table

ModuleCode	ModuleName	ModuleTeachingDays	ModuleCredit	
CC50NI	Application Development	24.00	30.00	Edit | Details | Delete
CC45NI	Database Management	26.00	30.00	Edit | Details | Delete
CS60NI	Work Related Learning	12.00	15.00	Edit | Details | Delete
CC13NI	Final Year Project	38.00	60.00	Edit | Details | Delete
CS14NI	Information System	12.00	30.00	Edit | Details | Delete

Figure 97: Module Table

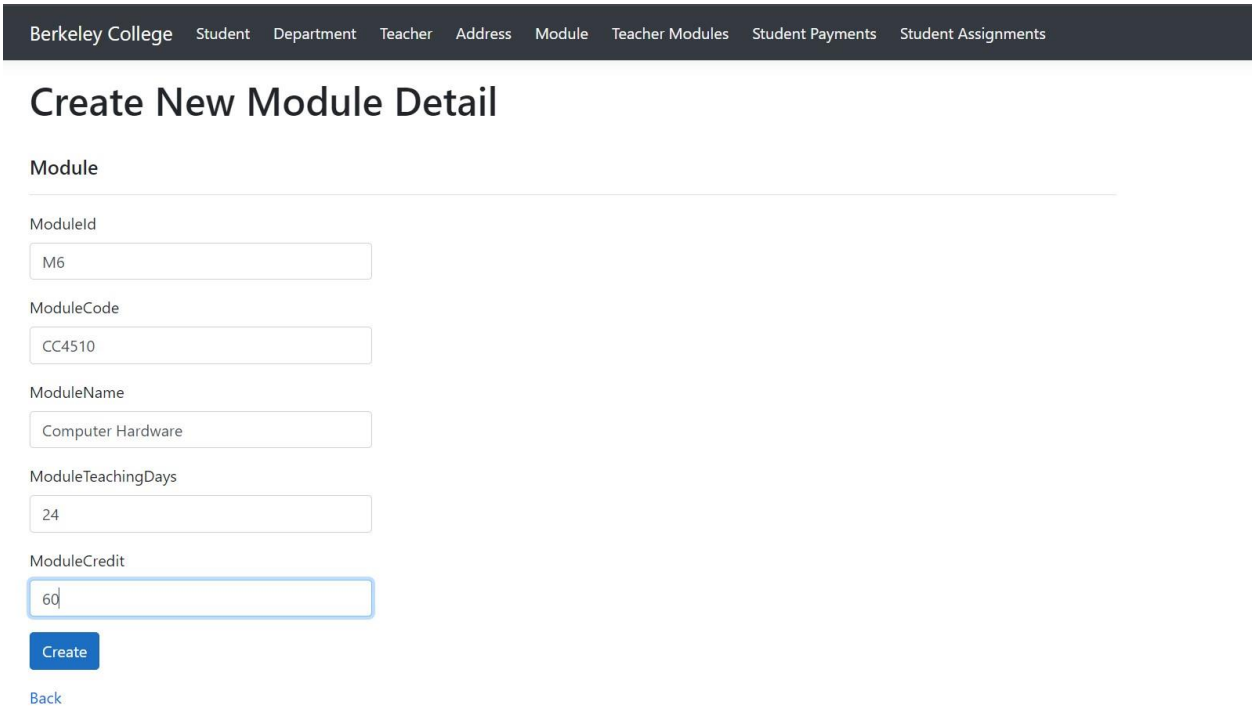


Figure 98: Inserting New Module Record

ModuleCode	ModuleName	ModuleTeachingDays	ModuleCredit	
CC50NI	Application Development	24.00	30.00	Edit | Details | Delete
CC45NI	Database Management	26.00	30.00	Edit | Details | Delete
CS60NI	Work Related Learning	12.00	15.00	Edit | Details | Delete
CC13NI	Final Year Project	38.00	60.00	Edit | Details | Delete
CS14NI	Information System	12.00	30.00	Edit | Details | Delete
CC4510	Computer Hardware	24.00	60.00	Edit | Details | Delete

Figure 99: Inserting New Module Record shown in table

Action	A new Module record is to be added in the Module table. Insert the records of the new Module and press Create button.
Expected Result	A new Module record is added in the table.
Actual Result	A new Module record was added in the table.
Conclusion	Test Successful

Table 25: Testing of create operation in Module Table

Update operation in Module Table

ModuleCode	ModuleName	ModuleTeachingDays	ModuleCredit	
CC50NI	Application Development	24.00	30.00	Edit | Details | Delete
CC45NI	Database Management	26.00	30.00	Edit | Details | Delete
CS60NI	Work Related Learning	12.00	15.00	Edit | Details | Delete
CC13NI	Final Year Project	38.00	60.00	Edit | Details | Delete
CS14NI	Information System	12.00	30.00	Edit | Details | Delete
CC4510	Computer Hardware	24.00	60.00	Edit | Details | Delete

Figure 100: Module Table

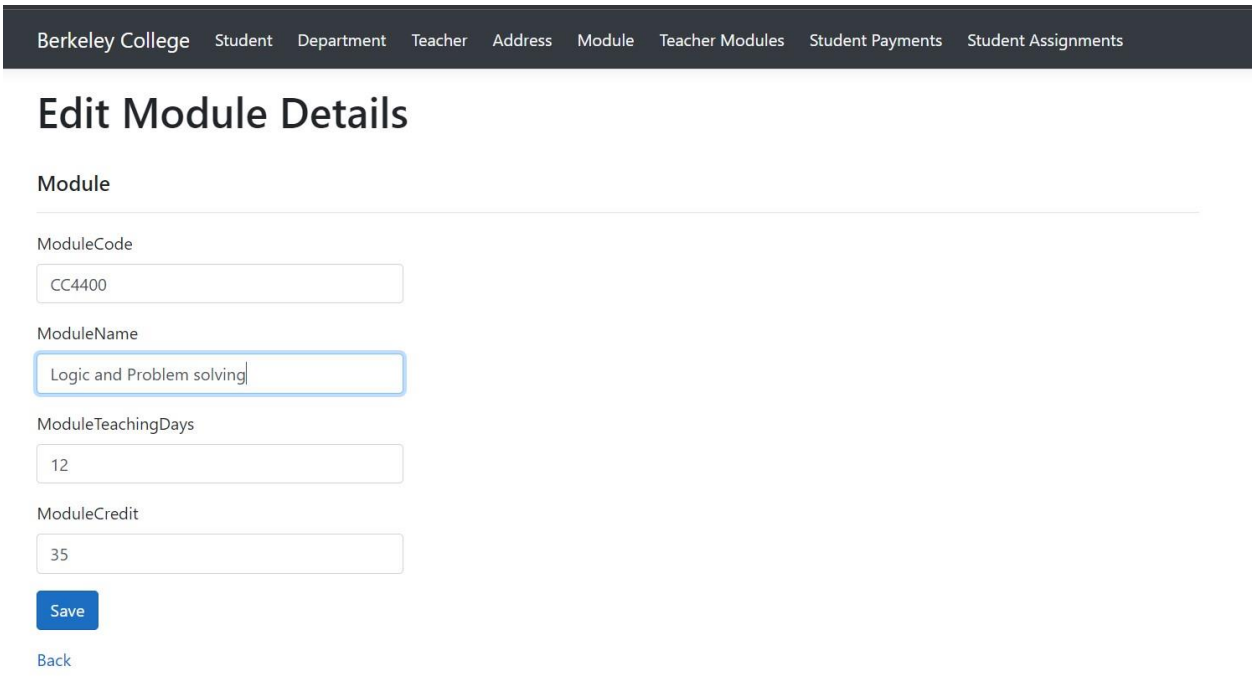


Figure 101: Updating a Module Record

ModuleCode	ModuleName	ModuleTeachingDays	ModuleCredit	
CC50NI	Application Development	24.00	30.00	Edit | Details | Delete
CC45NI	Database Management	26.00	30.00	Edit | Details | Delete
CS60NI	Work Related Learning	12.00	15.00	Edit | Details | Delete
CC13NI	Final Year Project	38.00	60.00	Edit | Details | Delete
CS14NI	Information System	12.00	30.00	Edit | Details | Delete
CC4400	Logic and Problem solving	12.00	35.00	Edit | Details | Delete

Figure 102: Updated Module Record shown in table

Action	A Module record is to be updated in the Module table. Insert the new records of the Module and press Edit button.
Expected Result	The Module record is updated in the table.
Actual Result	The Module record was updated in the table.
Conclusion	Test Successful

Table 26: Testing of update operation in Module Table

Delete operation in Module Table

ModuleCode	ModuleName	ModuleTeachingDays	ModuleCredit	
CC50NI	Application Development	24.00	30.00	Edit | Details | Delete
CC45NI	Database Management	26.00	30.00	Edit | Details | Delete
CS60NI	Work Related Learning	12.00	15.00	Edit | Details | Delete
CC13NI	Final Year Project	38.00	60.00	Edit | Details | Delete
CS14NI	Information System	12.00	30.00	Edit | Details | Delete
CC4400	Logic and Problem solving	12.00	35.00	Edit | Details | Delete

Figure 103 : Module Table before deleting a Module Record

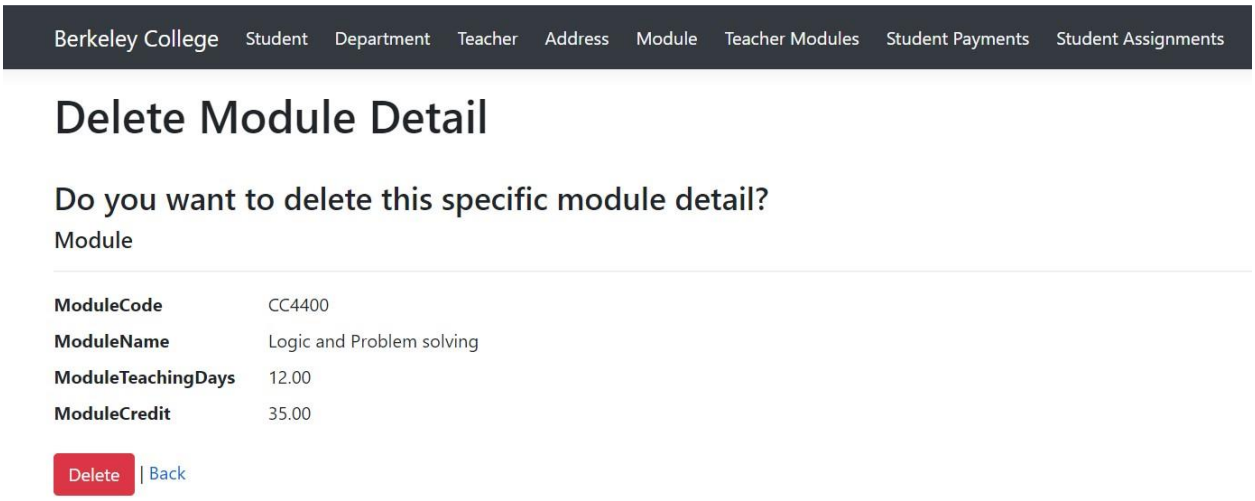


Figure 104: Deleting a Module Record

ModuleCode	ModuleName	ModuleTeachingDays	ModuleCredit	
CC50NI	Application Development	24.00	30.00	Edit | Details | Delete
CC45NI	Database Management	26.00	30.00	Edit | Details | Delete
CS60NI	Work Related Learning	12.00	15.00	Edit | Details | Delete
CC13NI	Final Year Project	38.00	60.00	Edit | Details | Delete
CS14NI	Information System	12.00	30.00	Edit | Details | Delete

Figure 105: Module Table after deleting a Module Record

Action	A Module record is to be deleted in the Module table. Press the Delete button in the Module record's row.
Expected Result	The Module record is deleted from the table.
Actual Result	The Module record was deleted from the table.
Conclusion	Test Successful

Table 27: Testing of delete operation in Module Table

Failed Cases

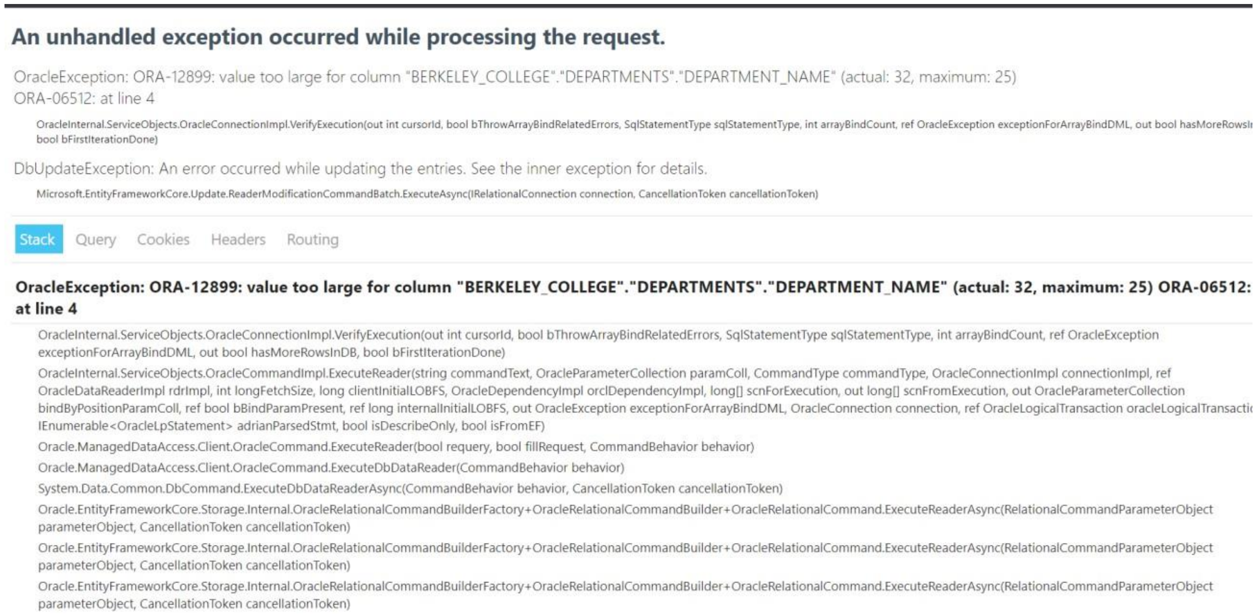


Figure 106: Failed cases

The above given diagram represents a failure in the code while I was inserting data in the departments table. The main reason for the occurrence of this error is that I exceeded the value length for column Department Name in the department table. For overcoming the error, I minimized the length of my department name string and inserted in the table. The error was solved quickly as I identified the error via terminal and corrected it

User Manual

User Manual for Student Table

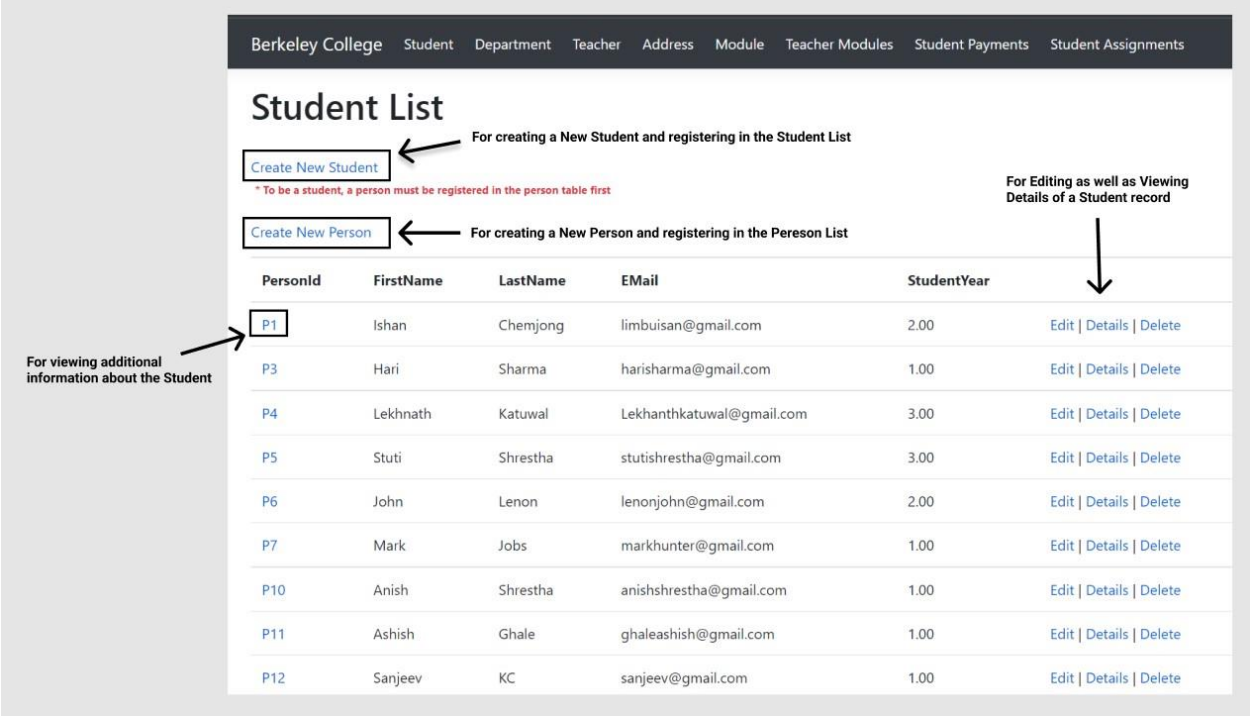


Figure 107: User Manual of Student Simple Form

A above given diagram is a graphical representation of a user manual of the Simple Student Form. The instructions and functions of every single option is listed below:

For Creating a New Student:

Step 1: For registering a new student, the person's data must be registered in the person table. If the person data is registered, simply click on the **Create New Student** Link.

Step 2: After that, the user will be redirected to the Create Page of Student Table. The user must fill up the necessary details of the Student and hit the **Create** Button With this, a new student record is created and registered in the Student Table.

For Editing a Student Record:

Step 1: For editing a student record, user should click on the **Edit** Button of the chosen student record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Student Table. The user must update the necessary changes in the Student Record and hit the **Edit** Button.

With this, the updated student record will show up in the Student Table.

For Viewing a Student Record:

Step 1: For viewing a student record, user should click on the **Details** Button of the chosen student record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Details Page of the Student table where additional information of the specific Student is listed.

For Deleting a Student Record:

Step 1: For deleting a student record, user should click on the **Delete** Button of the chosen student record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Student Table. The user should click on the **Delete** Button for deleting the student record.

With this, the deleted student record will be removed from the Student Table.

User Manual for Department Table

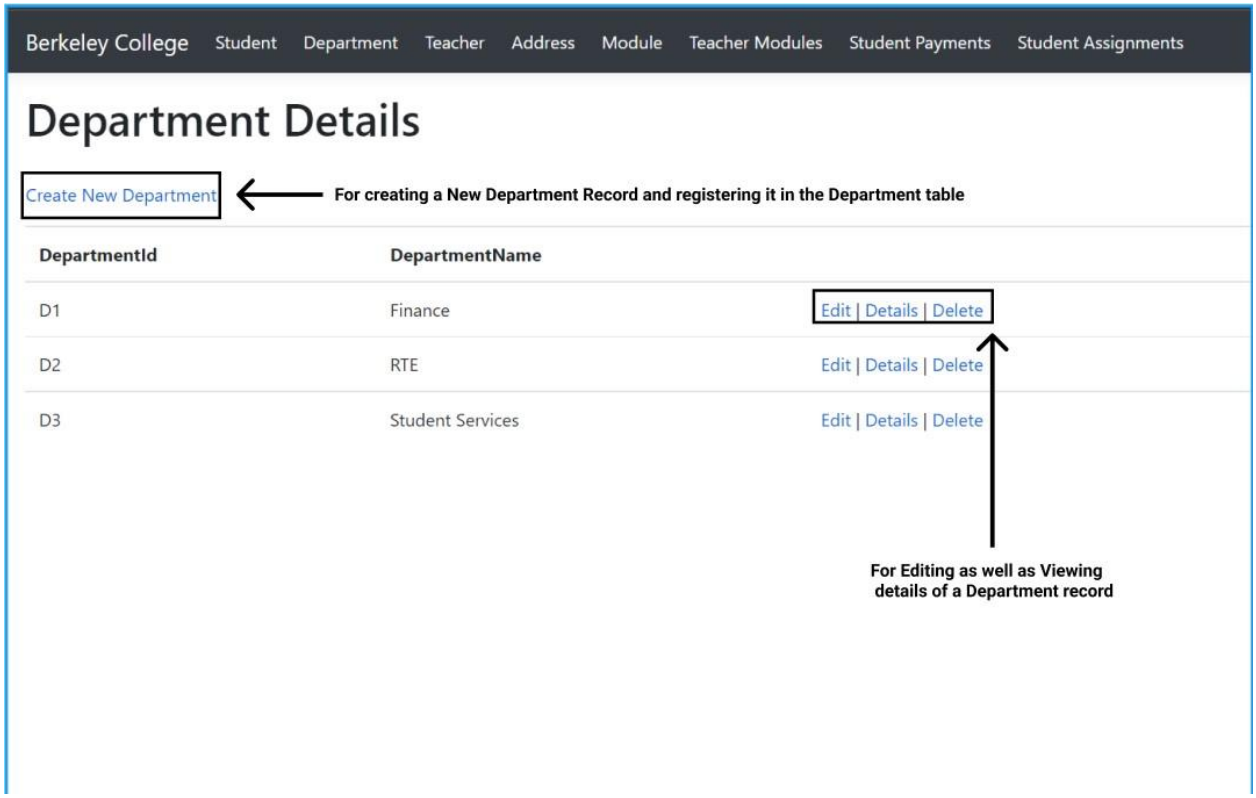


Figure 108: User Manual of Department Simple Form

A above given diagram is a graphical representation of a user manual of the Simple Department Form. The instructions and functions of every single option is listed below:

For Creating a New Department:

Step 1: For registering a new Department, simply click on the **Create New Department** Link.

Step 2: After that, the user will be redirected to the Create Page of Department Table. The user must fill up the necessary details of the Department and hit the **Create** Button

With this, a new Department record is created and registered in the Department Table.

For Editing a Department Record:

Step 1: For editing a Department record, user should click on the **Edit** Button of the chosen Department record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Department Table. The user must update the necessary changes in the Department Record and hit the **Edit** Button.

With this, the updated Department record will show up in the Department Table.

For Viewing a Department Record:

Step 1: For viewing a Department record, user should click on the **Details** Button of the chosen Department record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Details Page of the Department table where additional information of the specific Department is listed.

For Deleting a Department Record:

Step 1: For deleting a Department record, user should click on the **Delete** Button of the chosen Department record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Department Table. The user should click on the **Delete** Button for deleting the Department record.

With this, the deleted Department record will be removed from the Department Table.

User Manual for Teacher Table

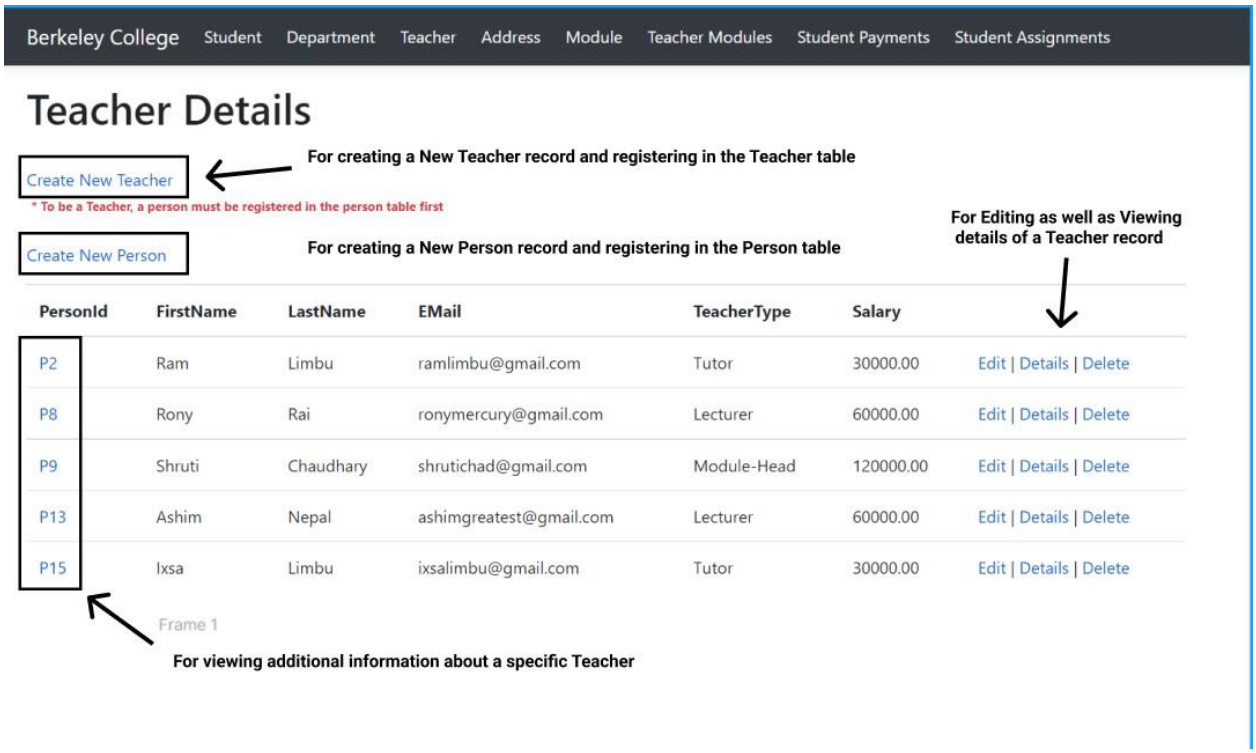


Figure 109: User Manual of Teacher Simple Form

A above given diagram is a graphical representation of a user manual of the Simple Teacher Form. The instructions and functions of every single option is listed below:

For Creating a New Teacher:

Step 1: For registering a new Teacher, the person's data must be registered in the person table. If the person data is registered, simply click on the **Create New Teacher** Link.

Step 2: After that, the user will be redirected to the Create Page of Teacher Table. The user must fill up the necessary details of the Teacher and hit the **Create** Button

With this, a new Teacher record is created and registered in the Teacher Table.

For Editing a Teacher Record:

Step 1: For editing a Teacher record, user should click on the **Edit** Button of the chosen Teacher record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Teacher Table. The user must update the necessary changes in the Teacher Record and hit the **Edit** Button.

With this, the updated Teacher record will show up in the Teacher Table.

For Viewing a Teacher Record:

Step 1: For viewing a Teacher record, user should click on the **Details** Button of the chosen Teacher record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Details Page of the Teacher table where additional information of the specific Teacher is listed.

For Deleting a Teacher Record:

Step 1: For deleting a Teacher record, user should click on the **Delete** Button of the chosen Teacher record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Teacher Table. The user should click on the **Delete** Button for deleting the Teacher record.

With this, the deleted Teacher record will be removed from the Teacher Table.

User Manual for Address Table

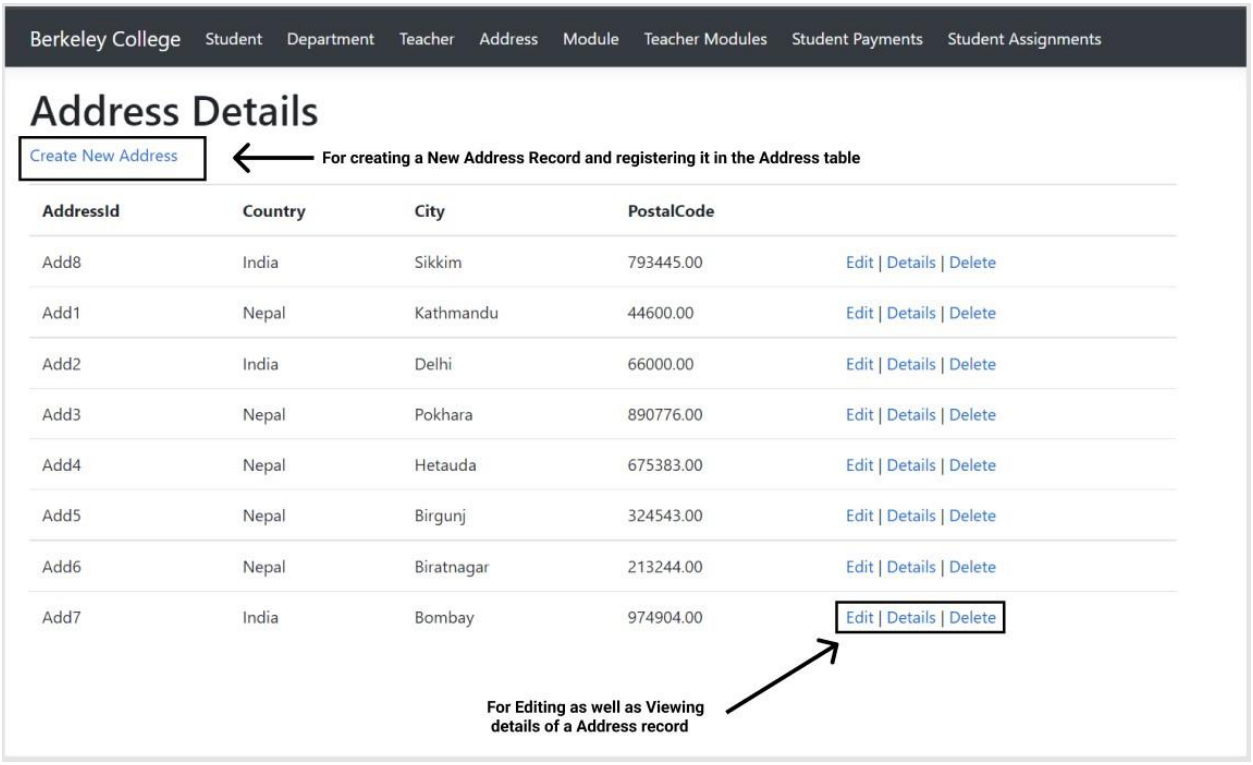


Figure 110: User Manual of Address Simple Form

A above given diagram is a graphical representation of a user manual of the Simple Address Form. The instructions and functions of every single option is listed below:

For Creating a New Address:

Step 1: For registering a new Address, simply click on the **Create New Address** Link.

Step 2: After that, the user will be redirected to the Create Page of Address Table. The user must fill up the necessary details of the Address and hit the **Create** Button

With this, a new Address record is created and registered in the Address Table.

For Editing an Address Record:

Step 1: For editing an Address record, user should click on the **Edit** Button of the chosen Address record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Address Table. The user must update the necessary changes in the Address Record and hit the **Edit** Button.

With this, the updated Address record will show up in the Address Table.

For Viewing an Address Record:

Step 1: For viewing an Address record, user should click on the **Details** Button of the chosen Address record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Details Page of the Address table where additional information of the specific Address is listed.

For Deleting an Address Record:

Step 1: For deleting an Address record, user should click on the **Delete** Button of the chosen Address record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Address Table. The user should click on the **Delete** Button for deleting the Address record.

With this, the deleted Address record will be removed from the Address Table.

User Manual for Module Table

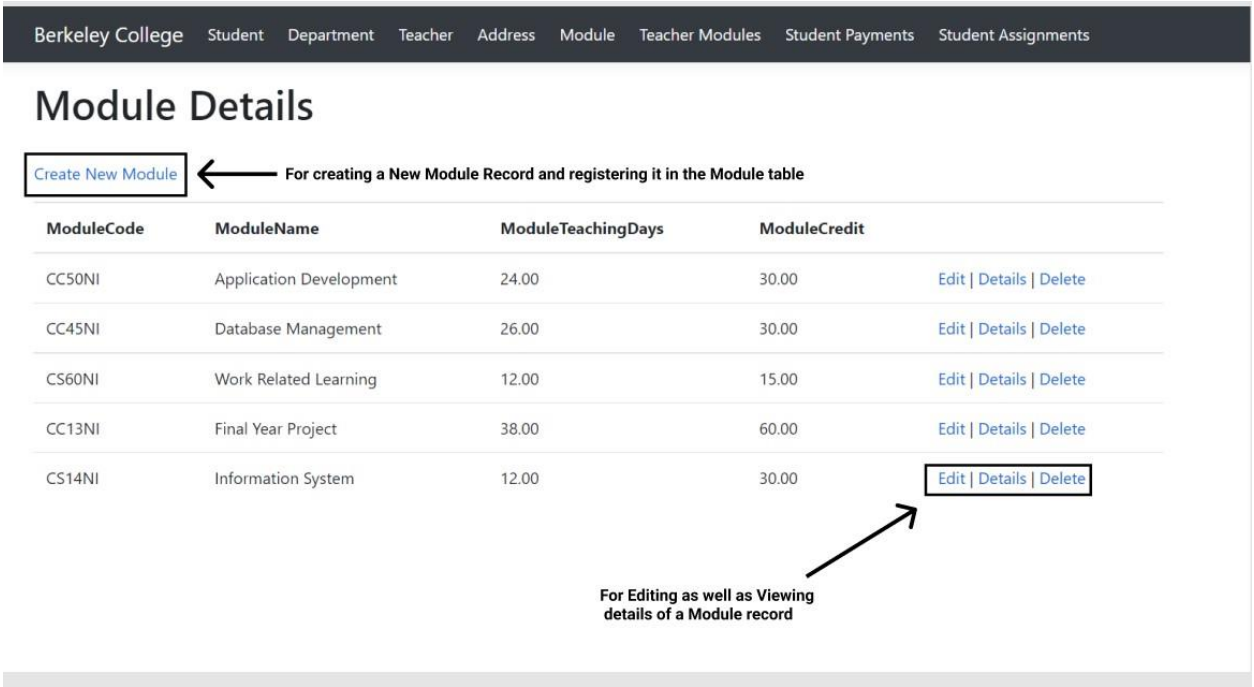


Figure 111: User Manual of Module Simple Form

A above given diagram is a graphical representation of a user manual of the Simple Module Form. The instructions and functions of every single option is listed below:

For Creating a New Module:

Step 1: For registering a new Module, simply click on the **Create New Module** Link.

Step 2: After that, the user will be redirected to the Create Page of Module Table. The user must fill up the necessary details of the Module and hit the **Create** Button

With this, a new Module record is created and registered in the Module Table.

For Editing a Module Record:

Step 1: For editing a Module record, user should click on the **Edit** Button of the chosen Module record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Module Table. The user must update the necessary changes in the Module Record and hit the **Edit** Button.

With this, the updated Module record will show up in the Module Table.

For Viewing a Module Record:

Step 1: For viewing a Module record, user should click on the **Details** Button of the chosen Module record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Details Page of the Module table where additional information of the specific Module is listed.

For Deleting a Module Record:

Step 1: For deleting a Module record, user should click on the **Delete** Button of the chosen Module record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Module Table. The user should click on the **Delete** Button for deleting the Module record.

With this, the deleted Module record will be removed from the Module Table.

User Manual for Teacher Module Mapping Form

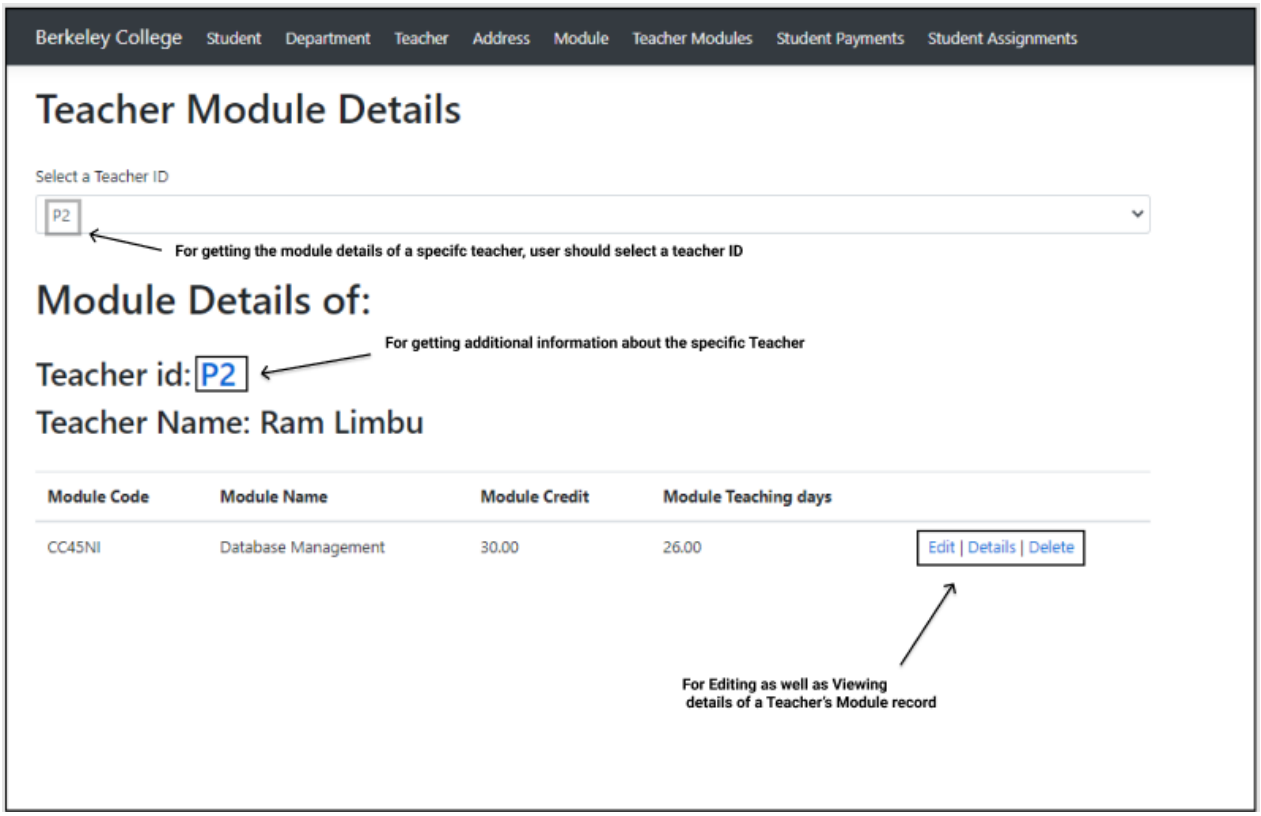


Figure 112: User Manual of Teacher Module Mapping Form

A above given diagram is a graphical representation of a user manual of the Teacher Module Mapping Form. The instructions and functions of every single option is listed below:

For Viewing a Teacher's Module Record:

Step 1: For viewing a Teacher's Module records, user should select a specific teacher Id from the select option in the form.

Step 2: After that, the user will be presented with a table enlisting all the modules taught by the specific teacher.

For Viewing a Teacher's Information:

Step 1: For viewing a Teacher's information, the user should click on **the Person ID** link of the teacher present in the form.

After that, the user will be redirected to the Details page of the specific teacher where all of the information is enlisted.

For Editing a Teacher's Module Record:

Step 1: For editing a Module record of a teacher, user should click on the **Edit** Button of the chosen Module record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Module Table. The user must update the necessary changes in the Module Record and hit the **Edit** Button.

With this, the updated Module record will show up in the Teacher Module Table.

For Deleting a Teacher's Module Record:

Step 1: For deleting a Module record of a teacher, user should click on the **Delete** Button of the chosen Module record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Module Table. The user should click on the **Delete** Button for deleting the Module record.

With this, the deleted Module record will be removed from the Teacher Module Table.

User Manual for Student Fee Payment Form

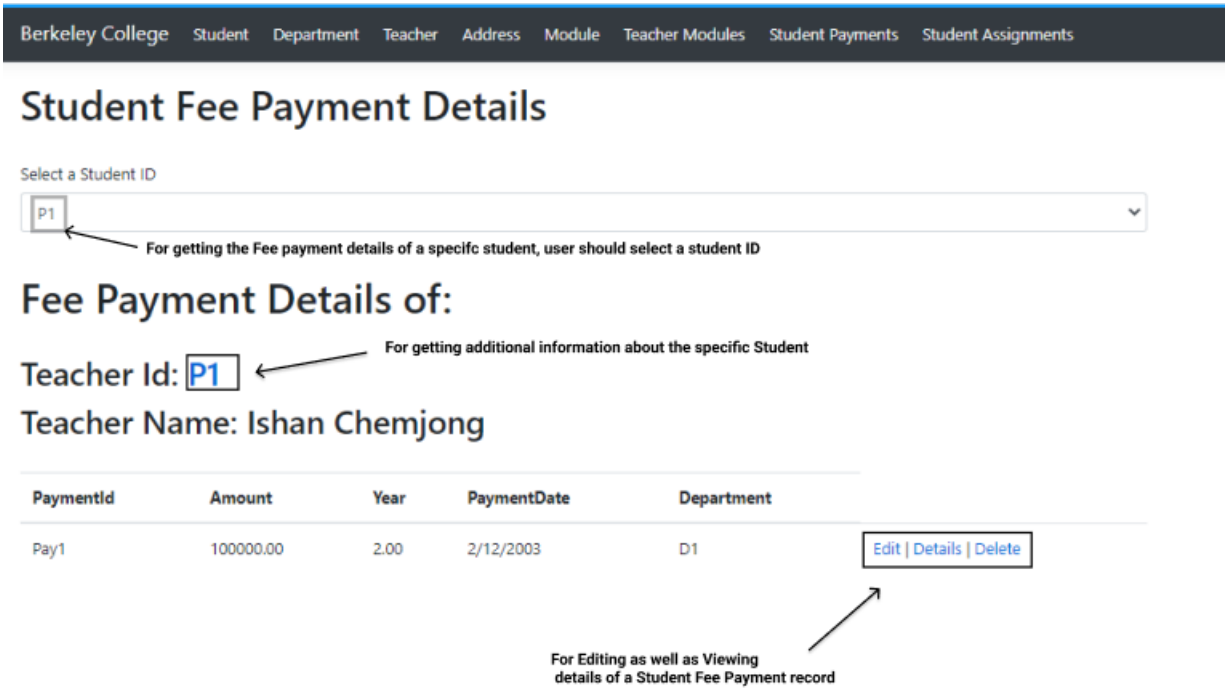


Figure 113: User Manual of Student Fee Payment Form

A above given diagram is a graphical representation of a user manual of the Student Fee Payment Form. The instructions and functions of every single option is listed below:

For Viewing a Student's Fee Payment Record:

Step 1: For viewing a Student's Fee Payment records, user should select a specific student Id from the select option in the form.

Step 2: After that, the user will be presented with a table enlisting all the payment records of the specific student.

For Viewing a Student's Information:

Step 1: For viewing a Student's information, the user should click on **the Person ID** link of the student present in the form.

After that, the user will be redirected to the Details page of the specific student where all the information is enlisted.

For Editing a Student's Fee Payment Record:

Step 1: For editing a Fee payment record of a student, user should click on the **Edit** Button of the chosen Payment record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Payment Table. The user must update the necessary changes in the Payment Record and hit the **Edit** Button.

With this, the updated Payment record will show up in the Student's Fee Payment Table.

For Deleting a Student's Fee Payment Record:

Step 1: For deleting a Fee payment of a student, user should click on the **Delete** Button of the chosen Fee payment record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Payment Table. The user should click on the **Delete** Button for deleting the Payment record.

With this, the deleted Payment record will be removed from the Student's Fee Payment Table.

User Manual for Student Assignment Form

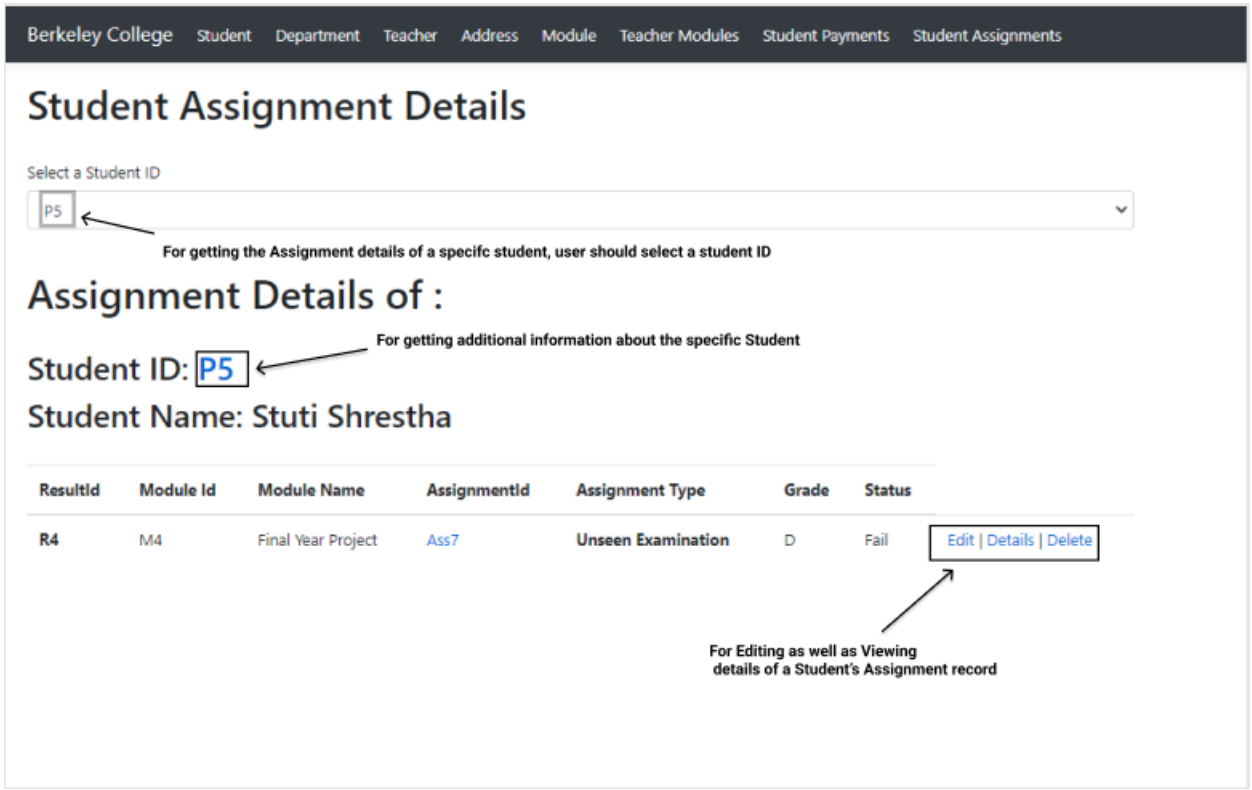


Figure 114: User Manual of Student Assignment Form

A above given diagram is a graphical representation of a user manual of the Student Assignment Form. The instructions and functions of every single option is listed below:

For Viewing a Student's Assignment Record:

Step 1: For viewing a Student's Assignment records, user should select a specific student Id from the select option in the form.

Step 2: After that, the user will be presented with a table enlisting all the Assignment records of the specific student.

For Viewing a Student's Information:

Step 1: For viewing a Student's information, the user should click on **the Person ID** link of the student present in the form.

After that, the user will be redirected to the Details page of the specific student where all the information is enlisted.

For Editing a Student's Assignment Record:

Step 1: For editing an Assignment record of a student, user should click on the **Edit** Button of the chosen Assignment record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Edit Page of Assignment Table. The user must update the necessary changes in the Assignment Record and hit the **Edit** Button.

With this, the updated Payment record will show up in the Student's Assignment Table.

For Deleting a Student's Assignment Record:

Step 1: For deleting an Assignment record of a student, user should click on the **Delete** Button of the chosen Assignment record which is present in the right-hand side of the table.

Step 2: After that, the user will be redirected to the Delete Page of Assignment Table. The user should click on the **Delete** Button for deleting the Assignment record.

With this, the deleted Assignment record will be removed from the Student's Assignment Table.

Further Discussion

During the construction of the web application as asked per the scenario of the coursework, I got an opportunity to use numerous applications and frameworks. I have used different tools and technologies such as SQL data modeler, SQL developer, Visual Studio and so on for constructing my web application. Firstly, I developed an intial ER-diagram of the database by the help of Draw.io as it is perfect for constructing small or skeleton diagrams. After that, I constructed the Final ER-model of my database with the help of SQL data modeler as it consists of different toolbars and drag drop features which reduces effort and time taken.

As the ER diagram had been finalized, I generated a DDL script of the diagram in the SQL developer which consisted of auto-generated create statements of the tables present in the diagram. Later, I executed the script in the SQL developer for constructing the tables in my database. SQL developer has made the generation and creation of database easier for me and helped me keep track of my execution code as well. I inserted all the values in the database tables by running the code again in the SQL developer. Till now, the database had been created.

Lastly, for the creation of the web application, I decided to implement Entity Framework Core 3.0 of ASP.NET as it is easy to use and keep track of the data. I used MVC architecture for manipulating data in my backend and used basic HTML, CSS and JS for the frontend part. All of this was done in Visual Studio Text editor as I found it very easy to use because of its debugging and other functions. For looking at the web application, I used Google Chrome Browser because of its simplicity.

With this, the entire coursework was completed, and the documentation was done in MS-Word as it provides a wide range of options that enables the user to beautify any sort of document. As I mentioned earlier, the construction of this web application is very important to me as it gave me chances to work with new tools and amplify my knowledge of database as well as application development.

Conclusion

The coursework's main objective was to develop a database and a web application through which we can view the data from the database and make changes to it. I have used ASP.NET for the backend of the web application and integrated it with the database. As the ASP.NET was a completely new to me, I had a hard time managing and integrating the code. For the data of each table, I had inserted fake information that is somewhat related to the actual college data. Apart from that many system errors as well as diagrammatical errors had occurred during the construction for the database. Our module tutor, Mr. Lekhnath Katuwal had helped me a lot overcome some of the challenges during the construction of the application. By the completion of this coursework, I have learned a lot about how a relational database works and integration between database and backend. The experience that I have gained by completing this project will certainly help me a lot in my future career.

References

FYICenter, 2011. [Online]

Available at: http://dba.fyicenter.com/faq/oracle/What-Is-Oracle-SQL-Developer.html

GeeksforGeeks, 2019. [Online]

Available at: https://www.geeksforgeeks.org/introduction-to-visual-studio/

GeeksforGeeks, 2020. [Online]

Available at: https://www.geeksforgeeks.org/types-of-normal-forms-in-dbms/

Oracle, 2022. [Online]

Available at: https://www.oracle.com/ph/database/technologies/appdev/datamodeler.html

Peterson, R., 2021. [Online]

Available at: https://www.guru99.com/database-normalization.html