



Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC6001NI - Advanced Database System Development

Assessment Weightage & Type

40% Individual Coursework

Year and Semester

2020-21 Autumn

Student Name: Suyogya Luitel

Group: C3

London Met ID: 19031784

College ID: NP01CP4A190035

Assignment Due Date: 23rd March 2022

Assignment Submission Date: 23rd March 2022

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

1.	Introduction	1
2.	Textual Analysis	2
2.1.	Teacher-Module	2
2.2.	Student-Module	2
2.3.	Student-Assignment	2
2.4.	Department-Fees-Assignment-Result	3
3.	ERD From Case Study	4
4.	Normalization	5
4.1.	Figure 1: Example of Teacher allocation list	5
4.1.1.	UNF	6
4.1.2.	1NF	6
4.1.3.	2NF	7
4.1.4.	3NF	9
4.1.5.	Final entities from figure 1	11
4.2.	Fig 2: Example of Assignment and Examination Results	12
4.2.1.	UNF	12
4.2.2.	1NF	13
4.2.3.	2NF	14
4.2.4.	3NF	17
4.2.5.	Final entities from figure 2	19
5.	Integration and Assumptions	20
6.	Final ERD	22
7.	Data Dictionary	23
7.1.	Table: Teacher	23

7.2. Table: Student.....	24
7.3. Table: Address	25
7.4. Table: Department	26
7.5. Table: Assignment	27
7.6. Table: Grade	28
7.7. Table: Module	29
7.8. Table: Module_Student	30
7.9. Table: Module_Teacher	31
7.10. Table: Semester.....	32
7.11. Table: Student_Assignment.....	33
7.12. Table: Student_Attendance	34
7.13. Table: Student_Fees.....	35
7.14. Table: Address_Teacher.....	37
8. DDL Script.....	38
8.1. Script.....	38
8.2. Output	48
9. INSERT Statements.....	54
9.1. Teacher.....	54
9.2. Student.....	55
9.3. Semester.....	56
9.4. Module	57
9.5. Grade	58
9.6. Department	59
9.7. Address	60
9.8. Assignment	61

9.9.	Student_Fees.....	62
9.10.	Student_Attendance	63
9.11.	Student_Assignment.....	64
9.12.	Module_Teacher.....	65
9.13.	Module_Student.....	66
9.14.	Address_Teacher.....	67
10.	SELECT Statements	68
10.1.	Address.....	68
10.2.	Address_Teacher.....	69
10.3.	Assignment.....	70
10.4.	Grade.....	71
10.5.	Module.....	72
10.6.	Module_Student.....	73
10.1.	Module_Teacher.....	74
10.1.	Semester	75
10.1.	Student	76
10.1.	Student_Assignment.....	77
10.1.	Student_Attendance	78
10.1.	Student_Fees	80
10.1.	Teacher.....	81
11.	Forms	82
11.1.	Dashboard	82
11.2.	Basic Forms.....	83
11.2.1.	Teacher Form.....	83
11.2.2.	Address Form.....	84

11.2.3.	Module Form	85
11.2.4.	Department Form	86
11.2.5.	Student Form.....	87
11.3.	Complex Forms.....	88
11.3.1.	Teacher-Module Mapping	88
11.4.	Student-Fees Mapping.....	91
11.5.	Student-Assignment Mapping	93
12.	User Manual	95
12.1.	Included Content.....	95
12.2.	Landing Page.....	96
12.3.	Teacher Form	97
12.4.	Address Form	98
12.5.	Module Form.....	99
12.6.	Department Form.....	100
12.7.	Student Form	101
12.8.	Teacher-Module Form	102
12.9.	Student-Fees Form	103
12.10.	Student-Assignment Form	104
13.	Testing.....	105
13.1.	Basic Forms	105
13.1.1.	Teacher Form.....	105
13.1.2.	Address Form.....	116
13.1.3.	Module Form	129
13.1.4.	Department Form	139
13.1.5.	Student Form.....	149

13.2. Complex Forms.....	158
13.2.1. Teacher-Module Mapping Form	158
13.2.2. Student-Fees Mapping Form.....	162
13.2.3. Student-Assignment Mapping Form	166
14. Further Discussion	168
15. Conclusion.....	169

Table of Figures

Figure 1: Relation between Module and Teacher.....	2
Figure 2: Relation between Module and Student	2
Figure 3: Relation between Student and Assignment	3
Figure 4: Relation between Department with Fees, Assignments and Results	3
Figure 5: Initial ERD from Case Study.....	4
Figure 6: Final ERD After Integration	22
Figure 7:DDL Script Output 1	48
Figure 8:DDL Script Output 2	49
Figure 9:DDL Script Output 3	50
Figure 10:DDL Script Output 4	51
Figure 11:DDL Script Output 5	52
Figure 12:DDL Script Output 6	53
Figure 13: INSERT Teacher.....	54
Figure 14: INSERT Student.....	55
Figure 15: INSERT Semester.....	56
Figure 16: INSERT Module	57
Figure 17: INSERT Grade	58
Figure 18: INSERT Department	59
Figure 19: INSERT Address	60
Figure 20: INSERT Assignment	61
Figure 21: INSERT Student_Fees.....	62
Figure 22: INSERT Student_Attendance.....	63
Figure 23: INSERT Student_Assignment.....	64
Figure 24: INSERT Module_Teacher	65
Figure 25: INSERT Module_Student.....	66
Figure 26: INSERT Address_Teacher.....	67
Figure 27:Select Address	68
Figure 28:Select Address_Teacher	69
Figure 29:Select Assignment	70
Figure 30:Select Grade	71

Figure 31:Select Module	72
Figure 32:Select Module_Student	73
Figure 33:Select Module_Teacher	74
Figure 34:Select Semester.....	75
Figure 35:Select Student.....	76
Figure 36:Select Student_Assignment	77
Figure 37:Select Attendance	78
Figure 38:Select Fees	80
Figure 39:Select Teacher	81
Figure 40: WebForms Dashboard	82
Figure 41: Teacher Form.....	83
Figure 42: Teacher Form.....	84
Figure 43: Module Form	85
Figure 44: Department Form	86
Figure 45: Student Form	87
Figure 46: Teacher Module Mapping Form Before Selection	88
Figure 47: Teacher Module Mapping Form After Selection	89
Figure 48: Student-Fees Mapping Form Before Selection	91
Figure 49: Student-Fees Mapping Form After Selection	91
Figure 50: Student-Assignment Mapping Form Before Selection.....	93
Figure 51: Student-Fees Mapping Form After Selection	93
Figure 52: Landing Page	96
Figure 53: Teacher Form.....	97
Figure 54: Address Form.....	98
Figure 55: Module Form	99
Figure 56: Department Form	100
Figure 57: Student Form	101
Figure 58: Teacher-Module Form.....	102
Figure 59: Student_Fees Form	103
Figure 60: Student_Assignment Form.....	104
Figure 61: Teacher Test 1 evidence.....	106

Figure 62: Teacher Test 2 Added Data	108
Figure 63: Teacher Test 2 Addition Result.....	109
Figure 64: Teacher Test 3 condition 1.....	111
Figure 65: Edited value in Email Textbox.....	112
Figure 66: Teacher Test 3 Condition 2.....	113
Figure 67: Teacher Grid Pre-Deletion	114
Figure 68: Teacher Grid Post-Deletion.....	115
Figure 69: Address Test 1 evidence.....	117
Figure 70: Address Test 2 Data to be Added	118
Figure 71: Address Test 2 Addition Result.....	119
Figure 72: Address Test 3 condition 1 fail case.....	121
Figure 73: Issue causing the fail case	122
Figure 74: Rectification of the Issue causing the fail case.....	122
Figure 75: Test 3 Condition 1 post rectification	123
Figure 76: Edited value in Street Name Textbox.....	125
Figure 77: Address Test 3 Condition 2	126
Figure 78: Address Grid Pre-Deletion	127
Figure 79: Address Grid Post-Deletion.....	128
Figure 80: Module Test 1 evidence	129
Figure 81: Module Test 2 Data to be Added.....	130
Figure 82: Module Test 2 Addition Result	131
Figure 83: Module Test 3 condition 1	133
Figure 84: Edited value in Credit Hours Textbox.....	134
Figure 85: Module Test 3 Condition 2	135
Figure 86: Address Grid Pre-Deletion	136
Figure 87: Issue causing the fail case	137
Figure 88: Rectification of the Issue causing the fail case.....	137
Figure 89: Test 4 post deletion.....	138
Figure 90: Department Test 1 evidence	139
Figure 91: Department Test 2 Added Data.....	141
Figure 92: Department Test 2 Addition Result	142

Figure 93: Department Test 3 condition 1	144
Figure 94: Edited value in Department Textbox	145
Figure 95: Department Test 3 Condition 2	146
Figure 96: Department Grid Pre-Deletion.....	147
Figure 97: Department Grid Post-Deletion	148
Figure 98: Student Test 1 evidence.....	149
Figure 99: Student Test 2 Added Data.....	150
Figure 100: Student Test 2 Addition Result.....	151
Figure 101: Student Test 3 condition 1.....	153
Figure 102: Edited value in Street Number Textbox.....	154
Figure 103: Student Test 3 Condition 2.....	155
Figure 104: Student Grid Pre-Deletion	156
Figure 105: Student Grid Post-Deletion.....	157
Figure 106: Teacher-Module Test 1 evidence	159
Figure 107: Teacher-Module Test 2 evidence	161
Figure 108: Student-Fees Test 1 evidence	163
Figure 109: Student-Fees Test 2 evidence	165
Figure 110: Student-Assignment Test 1 evidence.....	166
Figure 111: Student-Assignment Test 2 evidence.....	167

Table of Tables

Table 1: Example of Teacher allocation list.....	5
Table 2: Teacher Table Dictionary	23
Table 3: Student Table Dictionary	24
Table 4: Address Table Dictionary	25
Table 5: Department Table Dictionary.....	26
Table 6: Assignment Table Dictionary.....	27
Table 7: Grade Table Dictionary	28
Table 8: Module Table Dictionary.....	29
Table 9: Module_Student Table Dictionary	30
Table 10: Module_Teacher Table Dictionary.....	31
Table 11: Semester Table Dictionary	32
Table 12: Student_Assignment Table Dictionary	33
Table 13: Student_Attendance Table Dictionary	34
Table 14: Student_Attendance Table Dictionary	36
Table 15: Address_Teacher Table Dictionary	37
Table 16: Teacher Test 1	105
Table 17: Teacher Test 2	107
Table 18: Teacher Test 3	110
Table 19: Teacher Test 4	114
Table 20: Address Test 1	116
Table 21: Address Test 2	118
Table 22: Address Test 3 Fail Case	120
Table 23: Address Test 3 condition 2	124
Table 24: Address Test 4	127
Table 25: Module Test 1.....	129
Table 26: Module Test 2.....	130
Table 27: Module Test 3.....	132
Table 28: Module Test 4.....	136
Table 29: Department Test 1.....	139
Table 30: Department Test 2.....	140

Table 31: Department Test 3.....	143
Table 32: Department Test 4.....	147
Table 33: Student Test 1	149
Table 34: Student Test 2	150
Table 35: Student Test 3	152
Table 36: Student Test 4	156
Table 37: Teacher-Module Test 1	158
Table 38: Teacher-Module Test 2	160
Table 39: Student-Fees Test 1	162
Table 40: Student-Fees Test 2	164
Table 41: Student-Assignment Test 1	166
Table 42: Student-Assignment Test 2	167

1. Introduction

This is a report of the coursework involving analysis, design, and implementation of a web-based database application in accordance with the given business case study with the help of tools such as Oracle SQL Developer Data Modeler, Oracle SQL Developer and ASP.NET. According to the presented business scenario, the database should contain information regarding college departments; students, including fee status and attendance; assignments, and their results.

Initially the database is normalized based on given example data and integrated along with assumptions to better fit the given business scenario. The final entities and relations obtained from integration are visualized with Oracle Developer Data Modeler and a DDL script is generated. The generated DDL is then used to generate a database through Oracle SQL, which is then populated with sample data. Webforms are then created to perform CRUD operations on some tables and selective filtration operations on joined tables. Testing is then performed to make sure the system works fine.

2. Textual Analysis

From the given case study, I derived the following relations between entities:

2.1. Teacher-Module

A teacher is allowed to teach multiple modules.

A module can be taught by multiple teachers.

Hence, a many-to-many relation seems to be established.

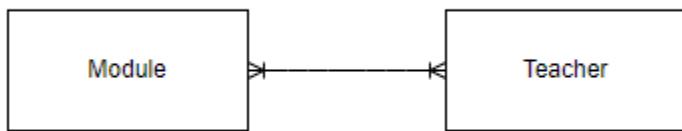


Figure 1: Relation between Module and Teacher

2.2. Student-Module

A student can study multiple modules.

A module can be studied by multiple students.

Hence a many-to-many relation seems to be established.

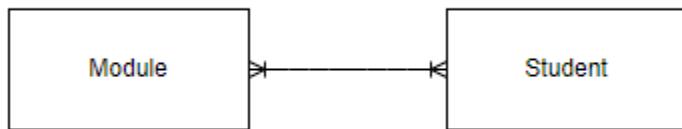


Figure 2: Relation between Module and Student

2.3. Student-Assessment

A student can partake in multiple assignments.

An assignment is submitted by multiple students.

Hence, a many-to-many relation is established.

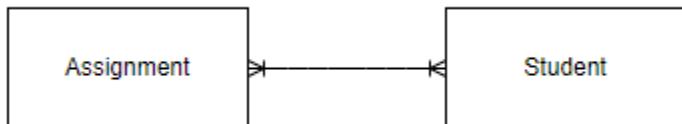


Figure 3: Relation between Student and Assignment

2.4. Department-Fees-Assignment-Result

A department manages multiple student fees.

A department manages multiple assignments.

A department manages multiple results.

Student fee is managed by a single department.

Assignment is managed by a single department.

Result is managed by a single department.

Hence, department establishes a one-to-many relation with fees, assignments, and results.

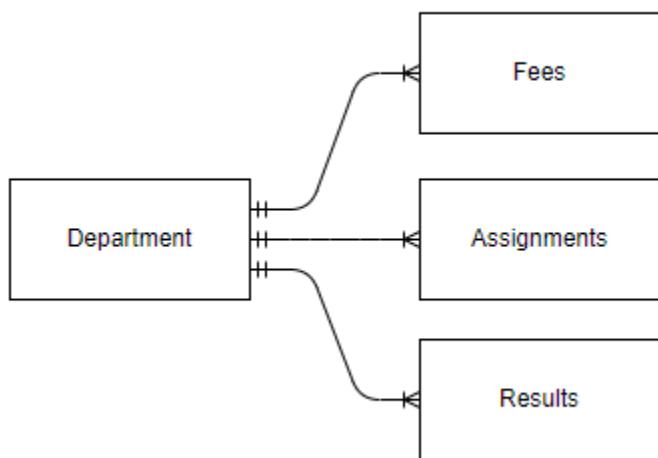


Figure 4: Relation between Department with Fees, Assignments and Results

3. ERD From Case Study

Based on the relations derived in the textual analysis of the given case study and some general assumptions, the following ERD is generated.

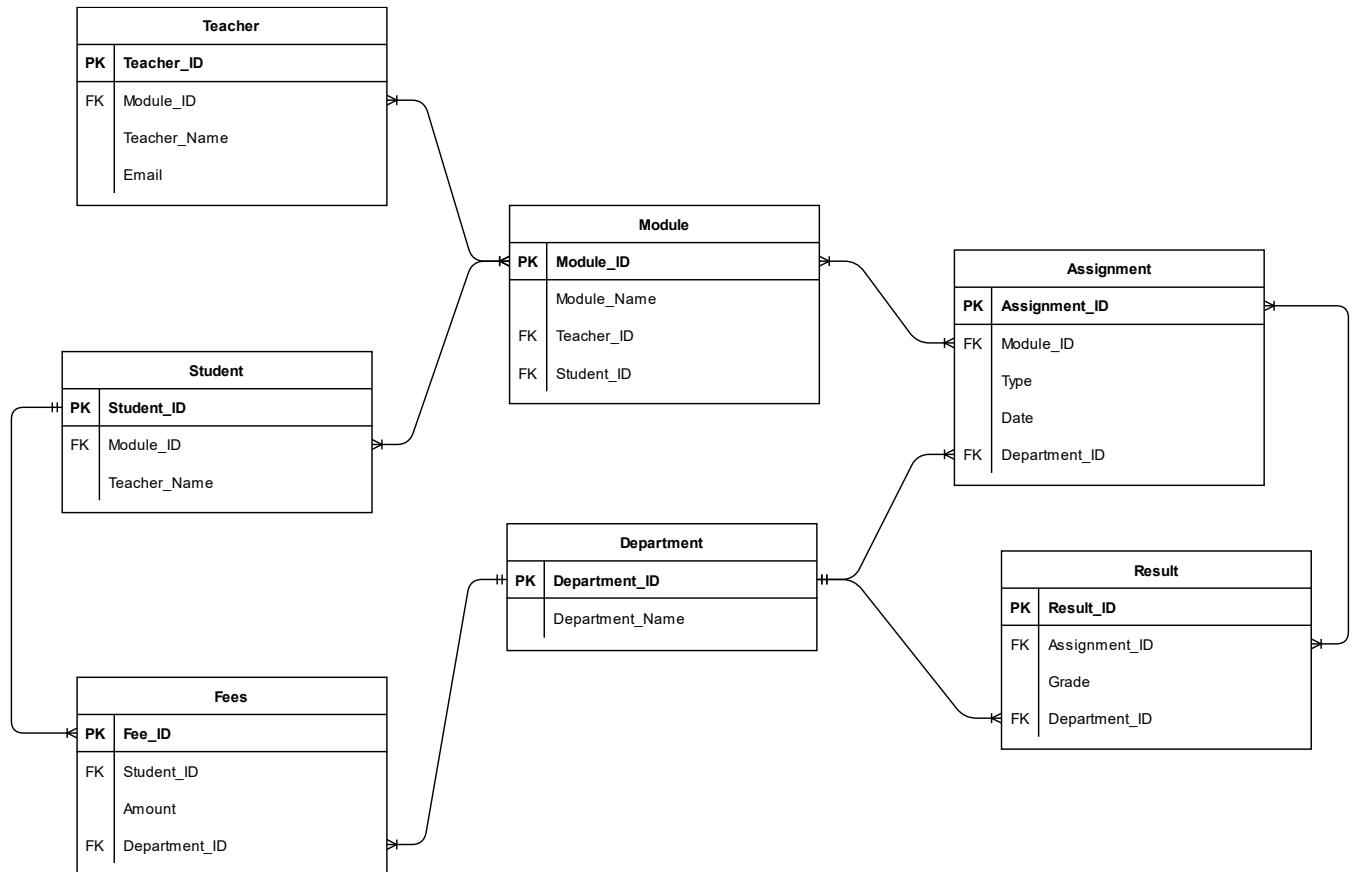


Figure 5: Initial ERD from Case Study

Assumptions:

- A student pays multiple fees throughout their stay in the college, and a fee belongs to only one student.
- Multiple results are published for the same assignment and a result relates to multiple assignments.
- A module may be included in multiple assignments and an assignment type may be implemented for multiple modules.

4. Normalization

4.1. Figure 1: Example of Teacher allocation list

S.N.	Teacher Name	Address		Email	Module Code	Module Name	Credit Hours
		1	2				
1	Saul Goodman	595 Green Lake Road Black Lake 9115 Lake Street Harrietsfield		Saulthegoodman@abc.edu.np	CC12	Data Structureand Algorithm	30
2	Walter White	696 Madison St. Pierrefonds		<u>whitywalker</u> @abc.e du.np	CC12	Data Structureand 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		<u>rustycohle</u> @abc.ed u.np	SG101	Softwareengineer	30
					TG405	Data Analysis	50

Table 1: Example of Teacher allocation list

Normalization of teacher allocation:

4.1.1. UNF

Identifying the repeating groups among the given figure fields, we get the following UNF:

Teacher (Teacher_ID, Teacher_Name, {Street_No., Street_Name, State_Name}, Email, {Module_Code, Module_Name, Credit_Hours,})

4.1.2. 1NF

Separating the repeating groups identified in the UNF above and assigning composite keys, we get the following entities:

Teacher-1 (Teacher_ID, Teacher_Name, Email)

Address-1 (Address_ID, Teacher_ID*, Street_No., Street_Name, State_Name)

Module-1 (Module_Code, Teacher_ID*, Module_Name, Credit_Hours)

4.1.3. 2NF

After the repeating groups were separated into different entities and assigned new composite keys, they needed to be checked for partial dependencies.

For Teacher-1:

- No composite keys were present and hence there were no partial dependencies.

$\text{Teacher_ID} \rightarrow \text{Teacher_Name, Email}$

$\text{Teacher-2 } (\underline{\text{Teacher_ID}}, \text{Teacher_Name, Email})$

For Address-1:

- Address_ID key determines $\text{Street_No., Street_Name, State_Name}$
- $\text{Address_ID, Teacher_ID}^*$ composite key determines nothing
- Teacher_ID^* foreign key determines nothing

$\text{Address_ID} \rightarrow (\text{Street_No., Street_Name, State_Name})$

$\text{Address_ID, Teacher_ID}^* \rightarrow ()$

$\text{Teacher_ID} \rightarrow ()$

The partial dependencies are separated into new entities as:

$\text{Address-2 } (\underline{\text{Address_ID}}^*, \text{Street_No., Street_Name, State_Name})$

$\text{Address-Teacher-2 } (\underline{\text{Address_ID}}, \underline{\text{Teacher_ID}}^*)$

For Module-1:

- Module_Code key determines $\text{Module_Name, Credit_Hours}$
- $\text{Module_Code, Teacher_ID}^*$ composite key determines nothing

$\text{Module_Code} \rightarrow (\text{Module_Name, Credit_Hours})$

Module_Code, Teacher_ID* → ()

Module -2 (Module_Code, Module_Name, Credit_Hours)

Module-Teacher -2 (Module_Code, Teacher_ID*)

Hence, the results of 2NF are the following entities:

Teacher-2 (Teacher_ID, Teacher_Name, Email)

Address-2 (Address_ID→ Street_No., Street_Name, State_Name)

Address-Teacher-2 (Address_ID, Teacher_ID*)

Module -2 (Module_Code, Module_Name, Credit_Hours)

Module-Teacher -2 (Module_Code, Teacher_ID*)

4.1.4. 3NF

After checking and removing any partial dependencies, transitive dependencies are required to be checked and removed.

For Teacher-2:

- Teacher_ID determines Teacher_Name and Email
- Teacher_Name determines nothing
- Email determines nothing

There are no transitive dependencies.

Teacher-3 (Teacher_ID, Teacher_Name, Email)

For Address-2:

- Address_ID determines Street_No., Street_Name, State_Name
- Address determines nothing

There are no transitive dependencies.

Address-3 (Address_ID, Street_No., Street_Name, State_Name)

For Address-Teacher 2:

There are no transitive dependencies, the entity contains only a composite primary key.

Address-Teacher-3 (Address_ID, Teacher_ID*)

For Module-Teacher 2:

There are no transitive dependencies, the entity contains only a composite primary key.

Module-Teacher-3 (Module_ID, Teacher_ID*)

For Module-2:

- Module_Code determines Module_Name, Credit_Hours
- Module_Name determines nothing
- Credit_Hours determines nothing

Module_Code → (Module_Name, Credit_Hours)

The transitive dependencies are separated into entities as follows:

Module-3 → (Module_Code, Module_Name, Credit_Hours)

Hence, the results of 3NF are the following entities:

Teacher-3 (Teacher_ID, Teacher_Name, Email)

Address-3 (Address_ID, Address)

Address-Teacher-3 (Address_ID, Teacher_ID*)

Module-3 (Module_Code, Module_Name, Credit_Hours)

Module-Teacher-3 (Module_ID, Teacher_ID*)

4.1.5. Final entities from figure 1

After normalizing figure 1 up to third normal form, the following entities are obtained:

Teacher (Teacher_ID, Teacher_Name, Email)

Address (Address_ID, Street_No., Street_Name, State_Name)

Address-Teacher (Address_ID, Teacher_ID*)

Module (Module_Code, Module_Name, Credit_Hours)

Module-Teacher (Module_ID, Teacher_ID*)

4.2. 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

Assumptions:

- A student cannot have multiple addresses.

4.2.1. UNF

Adding the above-mentioned fields to the given Figure 2 fields, and identifying the repeating groups among them, we get the following UNF:

Student (Student_ID, Sudent_Name, Street_No., Street_Name, State_Name, {Module_Code, Module_Name {Assignment_Type, Grade, Status}})

4.2.2. 1NF

Separating the repeating groups identified in the UNF above and assigning composite keys, we get the following entities:

Student-1 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

Module-1 (Module_Code, Student_ID*, Module_Name)

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

4.2.3. 2NF

After the repeating groups were separated into different entities and assigned new composite keys, they needed to be checked for partial dependencies.

For Student-1:

- No composite keys were present and hence there were no partial dependencies.

$\text{Student_ID} \rightarrow (\text{Student_Name}, \text{Street_No.}, \text{Street_Name}, \text{State_Name})$

Student-2 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

For Module-1:

- Module_Code key determines Module_Name
- Module_Code, Student_ID* composite key determines nothing
- Student_ID* foreign key determines nothing

$\text{Student_ID}, \text{Module_Code} \rightarrow ()$

$\text{Student_ID} \rightarrow ()$

$\text{Module_Code} \rightarrow (\text{Module_Name})$

Module-Student-2 (Student_ID*, Module_Code*)

Module-2 (Module_Code, Module_Name)

For Assignment-1:

- Assignment_ID key determines Assignment_Type
- Assignment_ID, Student_ID*, Module_Code* composite key determines Grade, Status
- Student_ID* foreign key determines nothing
- Module_Code* foreign key determines nothing

$\text{Assignment_ID} \rightarrow (\text{Assignment_Type})$

$\text{Assignment_ID}, \text{Student_ID}, \text{Module_Code} \rightarrow (\text{Grade}, \text{Status})$

$\text{Student_ID} \rightarrow ()$

$\text{Module_Code} \rightarrow ()$

The partial dependencies are separated into new entities as:

Assignment-2 (Assignment ID, Assignment Type)

Student-Assessment-2 (Assignment ID*, Student ID*, Module Code*, Grade, Status)

Hence, the results of 2NF are the following entities:

Student-2 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

Module-Student-2 (Student_ID*, Module_Code*)

Module-2 (Module_Code, Module_Name)

Assignment-2 (Assignment_ID, Assignment_Type)

Student-Assessment-2 (Assignment_ID*, Student_ID*, Module_Code*, Grade, Status)

4.2.4. 3NF

After checking and removing any partial dependencies, transitive dependencies are required to be checked and removed.

For Student-2 :

- Student_ID determines Student_Name, Street_No., Street_Name, State_Name
Student_Name determines nothing
- Student_Address determines nothing

$\text{Student_ID} \rightarrow (\text{Student_Name}, \text{Street_No.}, \text{Street_Name}, \text{State_Name})$

There are no transitive dependencies.

Student-3 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

For Module-Student-2:

There are no transitive dependencies, the entity contains only a composite primary key.

Module-Student-3 (Student_ID*, Module_Code*)

For Module-2:

- Module_Code determines Module_Name
- Module_Name determines nothing

There are no transitive dependencies.

Module-3 (Module_Code, Module_Name)

For Assignment-2:

- Assignment_ID determines Assignment_Type
- Assignment_Type determines nothing
- Module_Name determines nothing

There are no transitive dependencies.

Assignment-3 (Assignment_ID, Assignment_Type)

For Student-Assessment-2:

- Assignment_ID, Student_ID, Module_Code composite key determines Grade
- Grade determines Status
- Status determines nothing

Assignment_ID*, Student_ID*, Module_Code* → Grade

Grade → Status

The transitive dependencies are separated as:

Student-Assessment-3 (Assignment_ID*, Student_ID*, Module_Code*,
Grade_ID*)
Grade-3 (Grade_ID, Grade, Status)

Hence, the results of 3NF are the following entities :

Student-3 (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

Module-Student-3 (Student_ID*, Module_Code*)

Module-3 (Module_Code, Module_Name)

Assignment-3 (Assignment_ID, Assignment_Type)

Student-Assessment-3 (Assignment_ID*, Student_ID*, Module_Code*, Grade_ID*)

Grade-3 (Grade_ID, Grade, Status)

4.2.5. Final entities from figure 2

After normalizing figure 2 up to third normal form, the following entities are obtained:

Student (Student_ID, Student_Name, Street_No., Street_Name, State_Name)

Module-Student (Student_ID*, Module_Code*)

Module (Module_Code, Module_Name)

Assignment (Assignment_ID, Assignment_Type)

Student-Assignment (Assignment_ID*, Student_ID*, Module_Code*, Grade_ID*)

Grade (Grade_ID, Grade, Status)

5. Integration and Assumptions

Assumptions:

- A teacher may have multiple addresses, but a student can only have one address recorded.
- Grade is evaluated in a single alphabet optionally followed by a sign (Ex: A, A+).
- College has additional departments for managing examinations, assessments, results, as well as fees.
- Attendance of students is managed by a different system and recorded for each semester in percentage value in this database at the end of each semester right before examinations are conducted.
- Each semester has a fee associated with it which may or may not be equal for all semesters.
- Semester fees of the students are common regardless of the modules they study. This implies that students A and B studying module sets C and D have the same semester fees if they both are in semester E.
- Each semester, students are required to pay the allocated semester fees in a single installment.
- When a student pays semester fees, their fee status is updated to paid.
- To store the fees paid amount, the fee amount is redundantly added to the student_fees table so it always remains the same even if the semester fees change over time.
- Assignments and their results, attendance, and fees are managed by departments
- A student can study any three module they want.
- A teacher can teach one or multiple modules.
- Different semesters may have different fees.
- Student assignments for each module are recorded only after their grades have been evaluated.
- A student, after graduation, may become a teacher. Hence, a reference for student_id is added to the teacher table in case a student does become a teacher after graduation.

While the given example tables are very informative, they do not contain information regarding college departments. They also do not contain information regarding semester, semester fees, attendance, and fee status. Thus, the entities "Department", and "Semester", "Student_Fees" and "Student_Attendance" are added to the result of normalizing figures one and two with appropriate relations to existing entities.

Teacher (Teacher_ID, Teacher_Name, Email, Student_ID*)
Address (Address_ID, Street_No., Street_Name, State_Name)
Address-Teacher (Address_ID, Teacher_ID*)
Module (Module_Code, Module_Name, Credit_Hours)
Module-Teacher (Module_ID*, Teacher_ID*)
Student (Student_ID, Student_Name, Street_No., Street_Name, State_Name)
Module-Student (Student_ID*, Module_Code*)
Department (Department_ID, Department_Name)
Assignment (Assignment_ID, Assignment_Type, Department_ID*)
Student-Assignment (Assignment_ID*, Student_ID*, Module_Code*, Grade_ID*)
Grade (Grade_ID, Grade, Status)
Semester (Semester_ID, Semester, Semester_Fees)
Student_Fees (Semester_ID*, Student_ID*, Fee_Status, Department_ID*, Fee_Amount)
Student_Attendance (Semester_ID*, Student_ID*, Attendance_Percentage)

6. Final ERD

After normalization and integration of the given two example figures, the obtained entities form the following ERD:

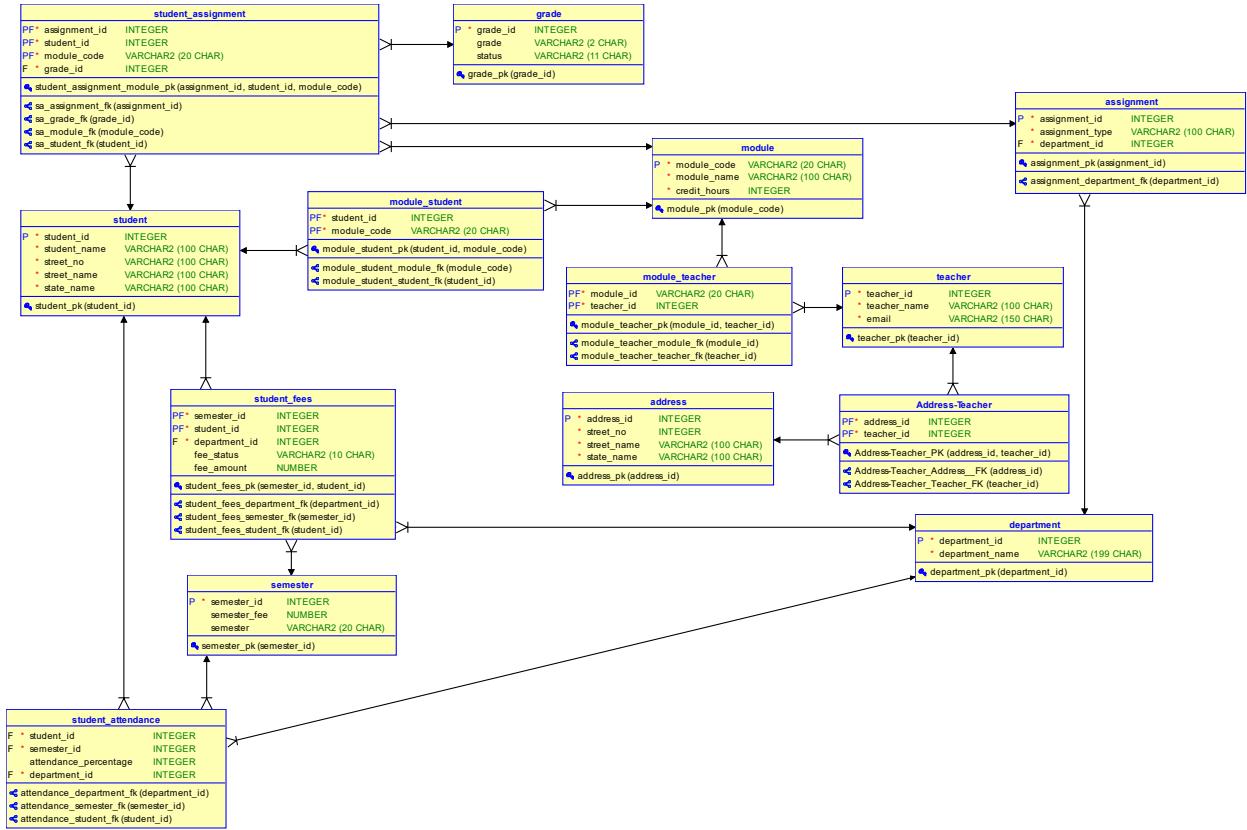


Figure 6: Final ERD After Integration

7. Data Dictionary

7.1.Table: Teacher

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Teacher_ID	INTEGER	38	Primary Key			Uniquely identifies each teacher	9
Teacher_Name	VARCHAR	100	NOT NULL			Stores the name of the teacher	4251
Email	VARCHAR	100	NOT NULL, UNIQUE			Stores the active email address of the teacher	Ethan_Cunningham 2529@brety.org
Student_ID	INTEGER	38				Stores student id of the person if they were a graduated student	6

Table 2: Teacher Table Dictionary

7.2.Table: Student

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	INTEGER	38	Primary Key			Uniquely identifies each student	5
Student_Name	VARCHAR	100	NOT NULL			Stores the name of the student	Ethan Shrestha
Street_No	INTEGER	38	NOT NULL			Stores street number of an address	4251
Street_Name	VARCHAR	100	NOT NULL			Stores street name of an address	Sheffield Walk
State_Name	VARCHAR	100	NOT NULL			Stores state name of an address	Hawaii

Table 3: Student Table Dictionary

7.3.Table: Address

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Address_ID	INTEGER	38	Primary Key			Uniquely identifies each address	3
Street_No	INTEGER	38	NOT NULL			Stores street number of an address	4251
Street_Name	VARCHAR	100	NOT NULL			Stores street name of an address	Sheffield Walk
State_Name	VARCHAR	100	NOT NULL			Stores state name of an address	Hawaii

Table 4: Address Table Dictionary

7.4. Table: Department

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Department_ID	INTEGER	38	Primary Key			Uniquely identifies each department	9
Department_Name	VARCHAR	100	NOT NULL, UNIQUE			Stores the name of the department	RTE

Table 5: Department Table Dictionary

7.5. Table: Assignment

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Assignment_ID	INTEGER	38	Primary Key			Uniquely identifies each assignment	2
Assignment_Type	VARCHAR	100	NOT NULL, UNIQUE			Stores the type of current assignment	4251
Department_ID	VARCHAR	100	NOT NULL	Department	Department_ID	Stores the department ID that manages assignments	Sheffield Walk

Table 6: Assignment Table Dictionary

7.6.Table: Grade

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Grade_ID	INTEGER	38	Primary Key			Uniquely identifies each grade as a number	2
Grade	VARCHAR	100	NOT NULL, UNIQUE			Stores the grade	D+
Status	VARCHAR	100	NOT NULL			Stores the pass or fail status of the grade	Pass

Table 7: Grade Table Dictionary

7.7. Table: Module

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Module_Code	VARCHAR	20	Primary Key			Uniquely identifies each assignment	CC12
Module_Name	VARCHAR	100	NOT NULL, UNIQUE			Stores the name of the module	Data Structure and Algorithm
Credit_Hours	INTEGER	38	NOT NULL			Stores the credit hours of the module	30

Table 8: Module Table Dictionary

7.8.Table: Module_Student

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	VARCHAR	100	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student associated with a module	6
Module_Code	VARCHAR	20	FOREIGN KEY, PRIMARY KEY (composite primary key)	Module	Module_Code	Stores the module code of a module associated with the student	CC12

Table 9: Module_Student Table Dictionary

7.9. Table: Module_Teacher

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Teacher_ID	VARCHAR	100	FOREIGN KEY, PRIMARY KEY (composite primary key)	Teacher	Teacher_ID	Stores the teacher id of the teacher associated with a module	9
Module_ID	VARCHAR	20	FOREIGN KEY, PRIMARY KEY (composite primary key)	Module	Module_Code	Stores the module code of a module associated with the teacher	CC12

Table 10: Module_Teacher Table Dictionary

7.10. Table: Semester

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Semester_ID	INTEGER	38	Primary Key			Uniquely identifies each semester	6
Semester	INTEGER	38	NOT NULL, UNIQUE			Stores the semester number	9
Semester_Fees	INTEGER	38	NOT NULL			Stores the fee for the semester	650

Table 11: Semester Table Dictionary

7.11. Table: Student_Assignment

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Assignment_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Assignment	Assignment_ID	Stores the assignment id of the assignment	6
Student_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student associated with the assignment	9
Module_Code	VARCHAR	20	FOREIGN KEY, PRIMARY KEY (composite primary key)	Module	Module_Code	Stores the module id of the module of which the assignment is	CC12
Grade_ID	INTEGER	38	FOREIGN KEY	Grade	Grade_ID	Stores the grade id of the grade obtained by the student	3

Table 12: Student_Assignment Table Dictionary

7.12. Table: Student_Attendance

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student whose attendance is being recorded	6
Semester_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Semester	Semester_ID	Stores the semester id of the semester in which the attendance was recorded	3
Department_ID	INTEGER	38	FOREIGN KEY, NOT NULL	Department	Department_ID	Stores the department id of the department responsible for managing attendance	1

Table 13: Student_Attendance Table Dictionary

7.13. Table: Student_Fees

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Student_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Student	Student_ID	Stores the student id of the student whose fee detail is being recorded	6
Semester_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Semester	Semester_ID	Stores the semester id of the semester for which the fee detail was recorded	3
Department_ID	INTEGER	38	FOREIGN KEY, NOT NULL	Department	Department_ID	Stores the department id of the department responsible for managing fees	2
Fee_Amount	NUMBER	(30,8)				Stores the amount of fee paid by the student	700

Fee_Status	VARCHAR	10				Stores the status of fees determining if the fee has been paid or not	Due
Payment_Date	DATE					Stores the date of payment of	TO_DATE('09, 01, 2017', 'MM, DD, YYYY')

Table 14: Student_Attendance Table Dictionary

7.14. Table: Address_Teacher

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Sample Data
Teacher_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Teacher	Teacher_ID	Stores the teacher id of the teacher associated with an address	6
Address_ID	INTEGER	38	FOREIGN KEY, PRIMARY KEY (composite primary key)	Address	Address_ID	Stores the address id of the address that is associated with the teacher	9

Table 15: Address_Teacher Table Dictionary

8. DDL Script

8.1. Script

```
-- Generated by Oracle SQL Developer Data Modeler 21.4.1.349.1605
-- at:      2022-03-12 18:05:35 NPT
-- site:    Oracle Database 11g
-- type:    Oracle Database 11g

-- predefined type, no DDL - MDSYS.SDO_GEOGRAPHY

-- predefined type, no DDL - XMLTYPE

CREATE SEQUENCE address_address_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE assignment_assignment_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE department_department_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE grade_grade_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE semester_semester_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE student_student_id_seq START WITH 1 NOCACHE ORDER;

CREATE SEQUENCE teacher_teacher_id_seq START WITH 1 NOCACHE ORDER;

CREATE TABLE address (
    address_id INTEGER NOT NULL,
    street_no INTEGER NOT NULL,
    street_name VARCHAR2(100 CHAR) NOT NULL,
    state_name VARCHAR2(100 CHAR) NOT NULL
)
;

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

CREATE TABLE address_teacher (
    address_id INTEGER NOT NULL,
    teacher_id INTEGER NOT NULL
)
;
```

```
ALTER TABLE address_teacher ADD CONSTRAINT "Address_Teacher_PK" PRIMARY KEY (
address_id,
t
eacher_id );

CREATE TABLE assignment (
    assignment_id    INTEGER NOT NULL,
    assignment_type VARCHAR2(100 CHAR) NOT NULL,
    department_id    INTEGER DEFAULT 1 NOT NULL
)
;

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

ALTER TABLE assignment ADD CONSTRAINT assignment_assignment_type_un UNIQUE (
assignment_type );

CREATE TABLE department (
    department_id    INTEGER NOT NULL,
    department_name VARCHAR2(199 CHAR) NOT NULL
)
;

ALTER TABLE department ADD CONSTRAINT department_pk PRIMARY KEY ( department_id
);
ALTER TABLE department ADD CONSTRAINT department_department_name_un UNIQUE (
department_name );

CREATE TABLE grade (
    grade_id    INTEGER NOT NULL,
    grade      VARCHAR2(2 CHAR),
    status     VARCHAR2(11 CHAR)
)
;

ALTER TABLE grade ADD CONSTRAINT grade_pk PRIMARY KEY ( grade_id );

ALTER TABLE grade ADD CONSTRAINT grade__un UNIQUE ( grade );

CREATE TABLE module (
    module_code   VARCHAR2(20 CHAR) NOT NULL,
    module_name   VARCHAR2(100 CHAR) NOT NULL,
    credit_hours INTEGER NOT NULL
)
```

```
;  
  
ALTER TABLE module ADD CONSTRAINT module_pk PRIMARY KEY ( module_code );  
  
ALTER TABLE module ADD CONSTRAINT module_module_name_un UNIQUE ( module_name );  
  
CREATE TABLE module_student (  
    student_id INTEGER NOT NULL,  
    module_code VARCHAR2(20 CHAR) NOT NULL  
)  
;  
  
ALTER TABLE module_student ADD CONSTRAINT module_student_pk PRIMARY KEY ( student_id,  
                                                                module_ code );  
  
CREATE TABLE module_teacher (  
    module_id VARCHAR2(20 CHAR) NOT NULL,  
    teacher_id INTEGER NOT NULL  
)  
;  
  
ALTER TABLE module_teacher ADD CONSTRAINT module_teacher_pk PRIMARY KEY ( module_id,  
                                                                teacher_id );  
  
CREATE TABLE semester (  
    semester_id INTEGER NOT NULL,  
    semester_fee NUMBER,  
    semester     VARCHAR2(20 CHAR)  
)  
;  
  
ALTER TABLE semester ADD CONSTRAINT semester_pk PRIMARY KEY ( semester_id );  
ALTER TABLE semester ADD CONSTRAINT semester_semester_un UNIQUE ( semester );  
  
CREATE TABLE student (  
    student_id   INTEGER NOT NULL,  
    student_name VARCHAR2(100 CHAR) NOT NULL,  
    street_no    VARCHAR2(100 CHAR) NOT NULL,  
    street_name   VARCHAR2(100 CHAR) NOT NULL,  
    state_name    VARCHAR2(100 CHAR) NOT NULL  
)
```

```
;  
  
ALTER TABLE student ADD CONSTRAINT student_pk PRIMARY KEY ( student_id );  
  
CREATE TABLE student_assignment (  
    assignment_id INTEGER NOT NULL,  
    student_id     INTEGER NOT NULL,  
    module_code    VARCHAR2(20 CHAR) NOT NULL,  
    grade_id      INTEGER NOT NULL  
)  
;  
  
ALTER TABLE student_assignment  
    ADD CONSTRAINT student_assignment_module_pk PRIMARY KEY ( assignment_id,  
                                                    student_id,  
                                                    module_code );  
  
CREATE TABLE student_attendance (  
    student_id          INTEGER NOT NULL,  
    semester_id         INTEGER NOT NULL,  
    attendance_percentage INTEGER,  
    department_id       INTEGER DEFAULT 1 NOT NULL  
)  
;  
  
ALTER TABLE student_attendance ADD CONSTRAINT student_attendance_pk PRIMARY KEY  
(student_id,  
semester_id);  
  
CREATE TABLE student_fees (  
    semester_id    INTEGER NOT NULL,  
    student_id     INTEGER NOT NULL,  
    department_id  INTEGER DEFAULT 2 NOT NULL,  
    fee_status     VARCHAR2(10 CHAR),  
    fee_amount     NUMBER(30,8),  
    payment_date   DATE  
)  
;  
  
ALTER TABLE student_fees ADD CONSTRAINT student_fees_pk PRIMARY KEY (  
semester_id,  
                                              student_id  
);
```

```
CREATE TABLE teacher (
    teacher_id    INTEGER NOT NULL,
    teacher_name  VARCHAR2(100 CHAR) NOT NULL,
    email         VARCHAR2(150 CHAR) NOT NULL,
    student_id    INTEGER DEFAULT NULL
)
;

ALTER TABLE teacher ADD CONSTRAINT teacher_pk PRIMARY KEY ( teacher_id );

ALTER TABLE teacher ADD CONSTRAINT teacher_email_un UNIQUE ( email );

ALTER TABLE address_teacher
    ADD CONSTRAINT "address_teacher_Address_FK" FOREIGN KEY ( address_id )
        REFERENCES address ( address_id )
            ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE address_teacher
    ADD CONSTRAINT "address_teacher_Teacher_FK" FOREIGN KEY ( teacher_id )
        REFERENCES teacher ( teacher_id )
            ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE assignment
    ADD CONSTRAINT assignment_department_fk FOREIGN KEY ( department_id )
        REFERENCES department ( department_id )
            ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_attendance
    ADD CONSTRAINT attendance_department_fk FOREIGN KEY ( department_id )
        REFERENCES department ( department_id )
            ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_attendance
    ADD CONSTRAINT attendance_semester_fk FOREIGN KEY ( semester_id )
        REFERENCES semester ( semester_id )
            ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_attendance
    ADD CONSTRAINT attendance_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
```

```
        ON DELETE CASCADE
NOT DEFERRABLE;

ALTER TABLE module_student
ADD CONSTRAINT module_student_module_fk FOREIGN KEY ( module_code )
    REFERENCES module ( module_code )
        ON DELETE CASCADE
NOT DEFERRABLE;

ALTER TABLE module_student
ADD CONSTRAINT module_student_student_fk FOREIGN KEY ( student_id )
    REFERENCES student ( student_id )
        ON DELETE CASCADE
NOT DEFERRABLE;

ALTER TABLE module_teacher
ADD CONSTRAINT module_teacher_module_fk FOREIGN KEY ( module_id )
    REFERENCES module ( module_code )
        ON DELETE CASCADE
NOT DEFERRABLE;

ALTER TABLE module_teacher
ADD CONSTRAINT module_teacher_teacher_fk FOREIGN KEY ( teacher_id )
    REFERENCES teacher ( teacher_id )
        ON DELETE CASCADE
NOT DEFERRABLE;

ALTER TABLE student_assignment
ADD CONSTRAINT sa_assignment_fk FOREIGN KEY ( assignment_id )
    REFERENCES assignment ( assignment_id )
        ON DELETE CASCADE
NOT DEFERRABLE;

ALTER TABLE student_assignment
ADD CONSTRAINT sa_grade_fk FOREIGN KEY ( grade_id )
    REFERENCES grade ( grade_id )
        ON DELETE CASCADE
NOT DEFERRABLE;

ALTER TABLE student_assignment
ADD CONSTRAINT sa_module_fk FOREIGN KEY ( module_code )
    REFERENCES module ( module_code )
        ON DELETE CASCADE
NOT DEFERRABLE;
```

```

ALTER TABLE student_assignment
    ADD CONSTRAINT sa_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
        ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_fees
    ADD CONSTRAINT student_fees_department_fk FOREIGN KEY ( department_id )
        REFERENCES department ( department_id )
        ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_fees
    ADD CONSTRAINT student_fees_semester_fk FOREIGN KEY ( semester_id )
        REFERENCES semester ( semester_id )
        ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE student_fees
    ADD CONSTRAINT student_fees_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
        ON DELETE CASCADE
    NOT DEFERRABLE;

ALTER TABLE teacher
    ADD CONSTRAINT teacher_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id )
        ON DELETE CASCADE
    NOT DEFERRABLE;

CREATE OR REPLACE TRIGGER address_address_id_trg BEFORE
    INSERT ON address
    FOR EACH ROW
    WHEN ( new.address_id IS NULL )
BEGIN
    :new.address_id := address_address_id_seq.nextval;
END;
/

CREATE OR REPLACE TRIGGER assignment_assignment_id_trg BEFORE
    INSERT ON assignment
    FOR EACH ROW
    WHEN ( new.assignment_id IS NULL )
BEGIN
    :new.assignment_id := assignment_assignment_id_seq.nextval;

```

```
END;
/

CREATE OR REPLACE TRIGGER department_department_id_trg BEFORE
    INSERT ON department
    FOR EACH ROW
    WHEN ( new.department_id IS NULL )
BEGIN
    :new.department_id := department_department_id_seq.nextval;
END;
/


CREATE OR REPLACE TRIGGER grade_grade_id_trg BEFORE
    INSERT ON grade
    FOR EACH ROW
    WHEN ( new.grade_id IS NULL )
BEGIN
    :new.grade_id := grade_grade_id_seq.nextval;
END;
/


CREATE OR REPLACE TRIGGER semester_semester_id_trg BEFORE
    INSERT ON semester
    FOR EACH ROW
    WHEN ( new.semester_id IS NULL )
BEGIN
    :new.semester_id := semester_semester_id_seq.nextval;
END;
/


CREATE OR REPLACE TRIGGER student_student_id_trg BEFORE
    INSERT ON student
    FOR EACH ROW
    WHEN ( new.student_id IS NULL )
BEGIN
    :new.student_id := student_student_id_seq.nextval;
END;
/


CREATE OR REPLACE TRIGGER teacher_teacher_id_trg BEFORE
    INSERT ON teacher
    FOR EACH ROW
    WHEN ( new.teacher_id IS NULL )
BEGIN
    :new.teacher_id := teacher_teacher_id_seq.nextval;

```

```
END;
/

-- Oracle SQL Developer Data Modeler Summary Report:

--
-- CREATE TABLE                      14
-- CREATE INDEX                       0
-- ALTER TABLE                        30
-- CREATE VIEW                         0
-- ALTER VIEW                          0
-- CREATE PACKAGE                      0
-- CREATE PACKAGE BODY                 0
-- CREATE PROCEDURE                    0
-- CREATE FUNCTION                     0
-- CREATE TRIGGER                      7
-- ALTER TRIGGER                       0
-- CREATE COLLECTION TYPE              0
-- CREATE STRUCTURED TYPE              0
-- CREATE STRUCTURED TYPE BODY         0
-- CREATE CLUSTER                      0
-- CREATE CONTEXT                      0
-- CREATE DATABASE                     0
-- CREATE DIMENSION                    0
-- CREATE DIRECTORY                    0
-- CREATE DISK GROUP                   0
-- CREATE ROLE                         0
-- CREATE ROLLBACK SEGMENT             0
-- CREATE SEQUENCE                     7
-- CREATE MATERIALIZED VIEW           0
-- CREATE MATERIALIZED VIEW LOG       0
-- CREATE SYNONYM                      0
-- CREATE TABLESPACE                   0
-- CREATE USER                         0
--

-- DROP TABLESPACE                    0
-- DROP DATABASE                      0
--

-- REDACTION POLICY                  0
--

-- ORDS DROP SCHEMA                  0
-- ORDS ENABLE SCHEMA                 0
-- ORDS ENABLE OBJECT                 0
--
```

```
-- ERRORS  
-- WARNINGS
```

```
0  
0
```

8.2. Output

The screenshot shows the Oracle SQL Developer interface with the 'Coursework' connection selected. The 'Script Output' tab is active, displaying the results of a DDL script execution. The output shows the creation of various database objects:

```
-- CREATE ROLLBACK SEGMENT
CREATE SEQUENCE
Sequence ADDRESS_ADDRESS_ID_SEQ created.

Sequence ASSIGNMENT_ASSIGNMENT_ID_SEQ created.

Sequence DEPARTMENT_DEPARTMENT_ID_SEQ created.

Sequence GRADE_GRADE_ID_SEQ created.

Sequence SEMESTER_SEMESTER_ID_SEQ created.

Sequence STUDENT_STUDENT_ID_SEQ created.

Sequence TEACHER_TEACHER_ID_SEQ created.

Table ADDRESS created.

Table ADDRESS altered.

Table ADDRESS_TEACHER created.

Table ADDRESS_TEACHER altered.

Table ASSIGNMENT created.

Table ASSIGNMENT altered.
```

Figure 7:DDL Script Output 1

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Coursework". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for file operations and database management. On the left, the Connections panel shows an Oracle connection named "Coursework" expanded to reveal tables, views, indexes, packages, procedures, functions, operators, queues, queues tables, triggers, types, sequences, materialized views, materialized view logs, synonyms, public synonyms, database links, public database links, and directories. Below it, the Reports panel lists all reports, analytic view reports, data dictionary reports, data modeler reports, OLAP reports, TimesTen reports, and user-defined reports. The central workspace consists of two tabs: "Worksheet" and "Script Output". The Worksheet tab displays a query builder interface with the following DDL statements:

```
-- CREATE ROLLBACK SEGMENT
-- CREATE SEQUENCE
```

The Script Output tab shows the execution results of the DDL statements:

```
Table ADDRESS_TEACHER altered.

Table ASSIGNMENT created.

Table ASSIGNMENT altered.

Table ASSIGNMENT altered.

Table DEPARTMENT created.

Table DEPARTMENT altered.

Table DEPARTMENT altered.

Table GRADE created.

Table GRADE altered.

Table GRADE altered.

Table MODULE created.

Table MODULE altered.

Table MODULE altered.
```

Figure 8: DDL Script Output 2

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Coursework". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for file operations like New, Open, Save, and Find.

The left sidebar has two sections: "Connections" and "Reports". Under "Connections", there is a tree view for "Coursework" containing Oracle Connections, Tables (Filtered), Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queue Tables, Triggers, Types, Sequences, Materialized Views, Materialized View Logs, Synonyms, Public Synonyms, Database Links, Public Database Links, and Directories. Under "Reports", there is a tree view for "All Reports" including Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports.

The main workspace consists of three panes: "Worksheet", "Query Builder", and "Script Output". The "Worksheet" pane shows the DDL script being run:

```
CREATE ROLLBACK SEGMENT
CREATE SEQUENCE
```

The "Script Output" pane displays the results of the executed DDL statements:

```
Table MODULE created.  
Table MODULE altered.  
Table MODULE altered.  
Table MODULE_STUDENT created.  
Table MODULE_STUDENT altered.  
Table MODULE_TEACHER created.  
Table MODULE_TEACHER altered.  
Table SEMESTER created.  
Table SEMESTER altered.  
Table SEMESTER altered.  
Table STUDENT created.  
Table STUDENT altered.  
Table STUDENT_ASSIGNMENT created.
```

Figure 9:DDL Script Output 3

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Coursework". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Print, and Database navigation.

The left side features two panels: "Connections" and "Reports". The "Connections" panel shows a tree view of "Coursework" database objects, including Tables (Filtered), Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queues Tables, Triggers, Types, Sequences, Materialized Views, Materialized View Logs, Synonyms, Public Synonyms, Database Links, Public Database Links, and Directories. The "Reports" panel lists All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports.

The main workspace displays the "Worksheet" tab with the following DDL script output:

```
-- CREATE ROLLBACK SEGMENT
CREATE SEQUENCE
Table STUDENT_ASSIGNMENT created.

Table STUDENT_ASSIGNMENT altered.

Table STUDENT_ATTENDANCE created.

Table STUDENT_ATTENDANCE altered.

Table STUDENT_FEES created.

Table STUDENT_FEES altered.

Table TEACHER created.

Table TEACHER altered.

Table TEACHER altered.

Table ADDRESS_TEACHER altered.

Table ADDRESS_TEACHER altered.

Table ASSIGNMENT altered.

Table STUDENT_ATTENDANCE altered.
```

Figure 10:DDL Script Output 4

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Coursework". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for connection management and SQL operations. On the left, there are two panes: "Connections" (showing a single connection named "Coursework" expanded to show Tables, Views, Indexes, etc.) and "Reports" (listing All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports). The central workspace consists of a "Worksheet" tab (selected) and a "Query Builder" tab. The Worksheet tab displays the following DDL script output:

```
-- CREATE ROLLBACK SEGMENT
CREATE SEQUENCE
Table STUDENT_ATTENDANCE altered.
Table STUDENT_ATTENDANCE altered.
Table STUDENT_ATTENDANCE altered.
Table MODULE_STUDENT altered.
Table MODULE_STUDENT altered.
Table MODULE_TEACHER altered.
Table MODULE_TEACHER altered.
Table STUDENT_ASSIGNMENT altered.
Table STUDENT_ASSIGNMENT altered.
Table STUDENT_ASSIGNMENT altered.
Table STUDENT_FEES altered.
Table STUDENT_FEES altered.
```

Figure 11:DDL Script Output 5

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which is expanded to show the 'Coursework' database connection, listing various schema objects like Tables, Views, Indexes, etc. Below it is the 'Reports' section. The main workspace consists of two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab contains the DDL script output. The script includes several 'CREATE' statements:

```
-- CREATE ROLLBACK SEGMENT
0
-- CREATE SEQUENCE
7

Table STUDENT_ASSIGNMENT altered.

Table STUDENT_FEES altered.

Table STUDENT_FEES altered.

Table STUDENT_FEES altered.

Table TEACHER altered.

Trigger ADDRESS_ADDRESS_ID_TRG compiled

Trigger ASSIGNMENT_ASSIGNMENT_ID_TRG compiled

Trigger DEPARTMENT_DEPARTMENT_ID_TRG compiled

Trigger GRADE_GRADE_ID_TRG compiled

Trigger SEMESTER_SEMESTER_ID_TRG compiled

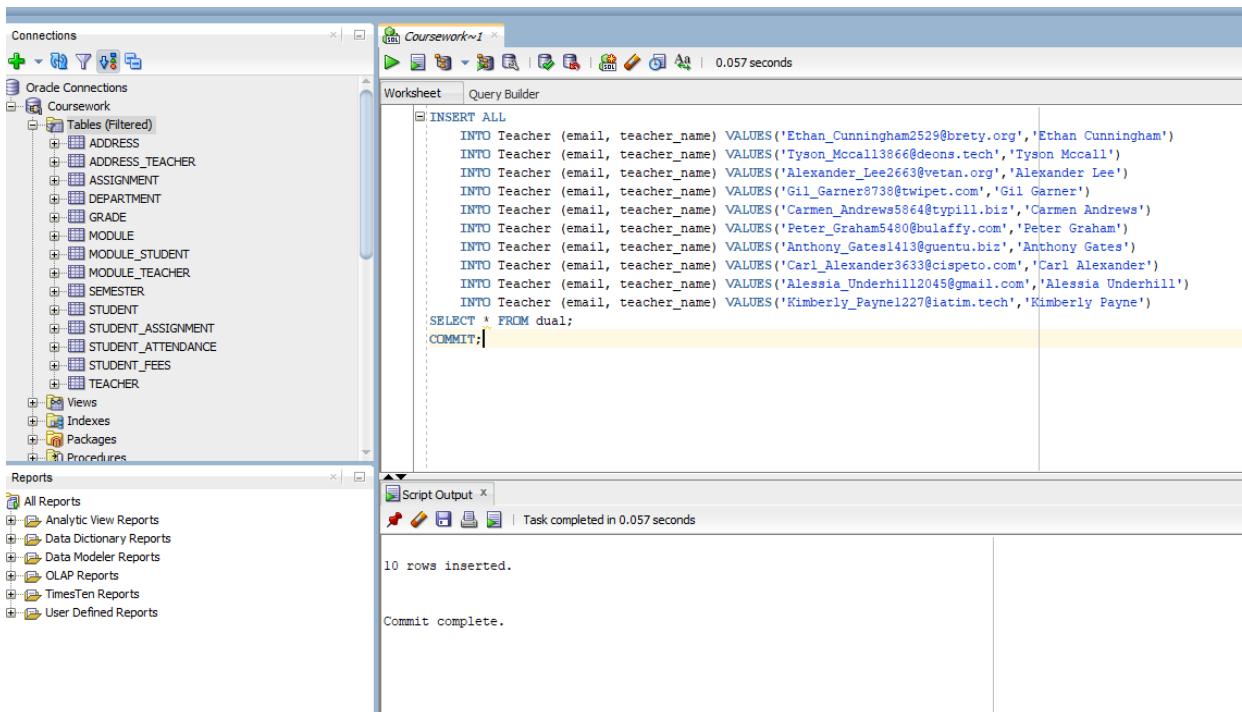
Trigger STUDENT_STUDENT_ID_TRG compiled

Trigger TEACHER_TEACHER_ID_TRG compiled
```

Figure 12:DDL Script Output 6

9. INSERT Statements

9.1. Teacher



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with tables like ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, and TEACHER. The central workspace contains a query editor window titled 'Coursework~1' with the following SQL code:

```

INSERT ALL
  INTO Teacher (email, teacher_name) VALUES('Ethan_Cunningham2529@brety.org', 'Ethan Cunningham')
  INTO Teacher (email, teacher_name) VALUES('Tyson_Mccall13866@deons.tech', 'Tyson McCall')
  INTO Teacher (email, teacher_name) VALUES('Alexander_Lee2663@vetan.org', 'Alexander Lee')
  INTO Teacher (email, teacher_name) VALUES('Gil_Garner8738@twipet.com', 'Gil Garner')
  INTO Teacher (email, teacher_name) VALUES('Carmen_Andrews5864@typill.biz', 'Carmen Andrews')
  INTO Teacher (email, teacher_name) VALUES('Peter_Graham5480@bulaffy.com', 'Peter Graham')
  INTO Teacher (email, teacher_name) VALUES('Anthony_Gates1413@guentu.biz', 'Anthony Gates')
  INTO Teacher (email, teacher_name) VALUES('Carl_Alexander3633@cispeto.com', 'Carl Alexander')
  INTO Teacher (email, teacher_name) VALUES('Alessia_Underhill2045@gmail.com', 'Alessia Underhill')
  INTO Teacher (email, teacher_name) VALUES('Kimberly_Payne1227@atim.tech', 'Kimberly Payne')
SELECT * FROM dual;
COMMIT;

```

The bottom right pane, 'Script Output', shows the results of the execution:

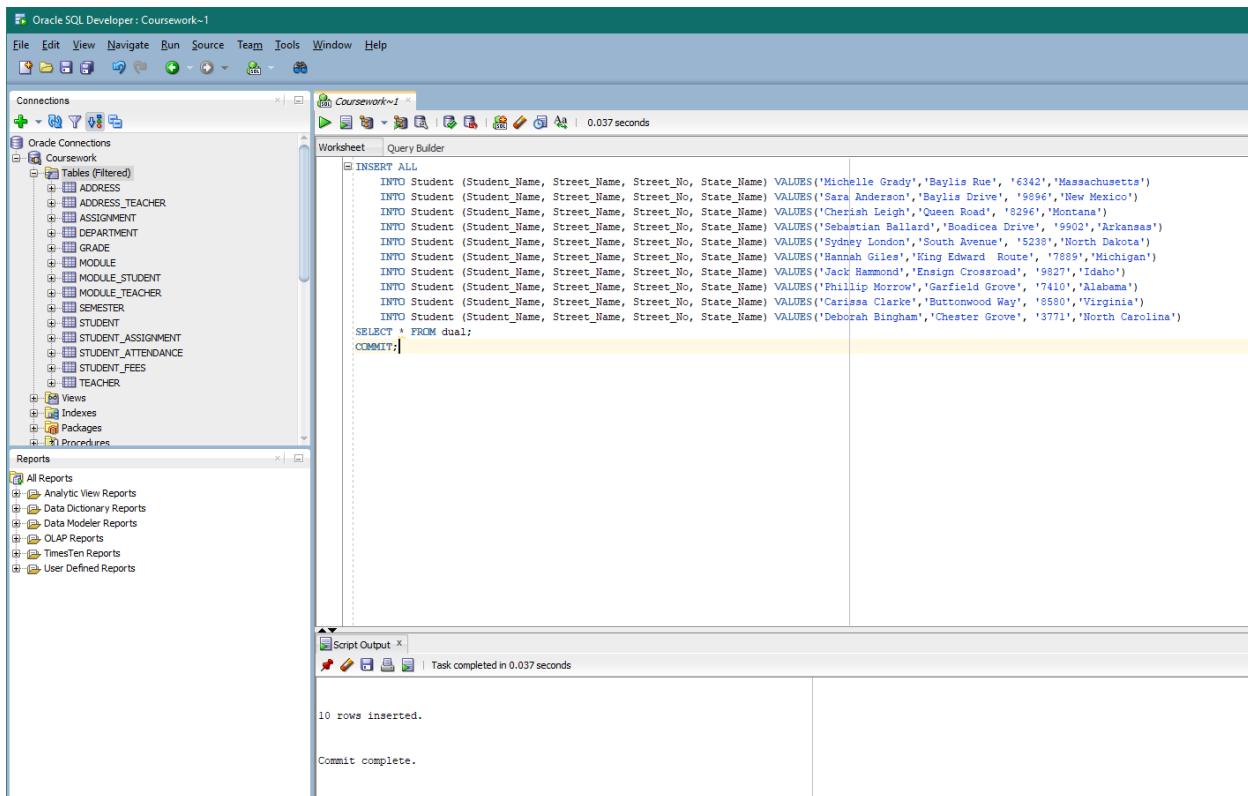
```

Task completed in 0.057 seconds
10 rows inserted.
Commit complete.

```

Figure 13: INSERT Teacher

9.2. Student



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, Coursework.
- Worksheet:** Query Builder tab, showing the following SQL code:


```
INSERT ALL
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Michelle Grady', 'Baylis Rue', '6342', 'Massachusetts')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Sara Anderson', 'Baylis Drive', '6896', 'New Mexico')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Cherish Leigh', 'Queen Road', '8256', 'Montana')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Sebastian Ballard', 'Boadicea Drive', '9802', 'Arkansas')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Sydney London', 'South Avenue', '5238', 'North Dakota')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Hannah Giles', 'King Edward Route', '7889', 'Michigan')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Jack Hammond', 'Ensign Crossroad', '9827', 'Idaho')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Phillip Morrow', 'Garfield Grove', '7410', 'Alabama')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Carissa Clarke', 'Buttonwood Way', '8580', 'Virginia')
      INTO Student (Student_Name, Street_Name, Street_No, State_Name) VALUES ('Deborah Bingham', 'Chester Grove', '3771', 'North Carolina')
      SELECT * FROM dual;
      COMMIT;
```
- Script Output:** Task completed in 0.037 seconds. The output shows:


```
10 rows inserted.

Commit complete.
```

Figure 14: INSERT Student

9.3. Semester

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar on the left lists 'Coursework' under 'Oracle Connections'. The 'Tables (Filtered)' section within 'Coursework' contains entries for ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, and TEACHER. The 'Reports' sidebar lists various report types. The central 'Worksheet' tab displays an SQL script:

```
INSERT ALL
  INTO Semester (Semester_Fee, Semester) VALUES (650, 1)
  INTO Semester (Semester_Fee, Semester) VALUES (550, 2)
  INTO Semester (Semester_Fee, Semester) VALUES (600, 3)
  INTO Semester (Semester_Fee, Semester) VALUES (650, 4)
  INTO Semester (Semester_Fee, Semester) VALUES (700, 5)
  INTO Semester (Semester_Fee, Semester) VALUES (750, 6)
SELECT * FROM dual;
COMMIT;
```

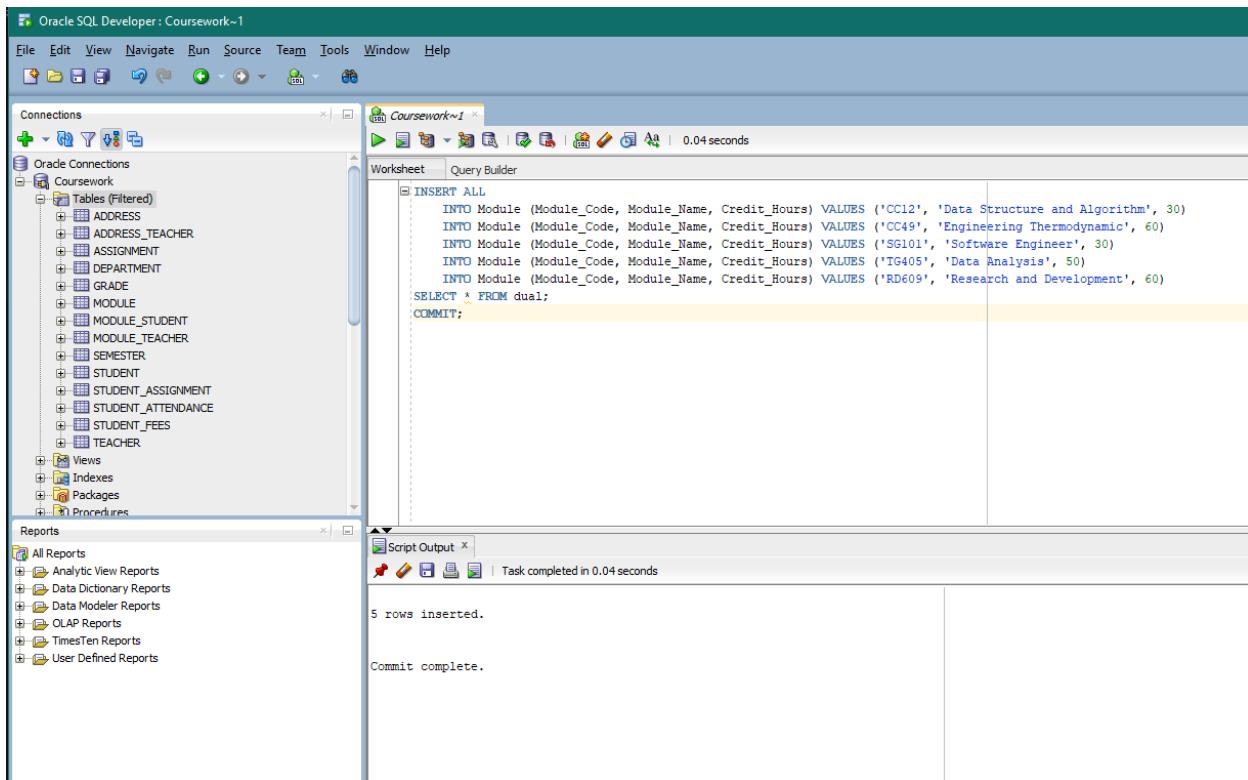
The 'Script Output' tab at the bottom shows the results of the execution:

```
6 rows inserted.

Commit complete.
```

Figure 15: INSERT Semester

9.4. Module



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'Oracle Connections' and 'Coursework'. Under 'Coursework', there are several tables listed: ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, and TEACHER. Below the connections tree are sections for 'Views', 'Indexes', 'Packages', and 'Procedures'. The main workspace is titled 'Coursework~1' and contains a 'Worksheet' tab with a 'Query Builder' sub-tab. The query editor displays the following SQL script:

```
INSERT ALL
  INTO Module (Module_Code, Module_Name, Credit_Hours) VALUES ('CC12', 'Data Structure and Algorithm', 30)
  INTO Module (Module_Code, Module_Name, Credit_Hours) VALUES ('CC49', 'Engineering Thermodynamic', 60)
  INTO Module (Module_Code, Module_Name, Credit_Hours) VALUES ('SG101', 'Software Engineer', 30)
  INTO Module (Module_Code, Module_Name, Credit_Hours) VALUES ('TG405', 'Data Analysis', 50)
  INTO Module (Module_Code, Module_Name, Credit_Hours) VALUES ('RD609', 'Research and Development', 60)
SELECT * FROM dual;
COMMIT;
```

Below the worksheet is a 'Script Output' window showing the results of the execution:

```
5 rows inserted.  
Commit complete.
```

Figure 16: INSERT Module

9.5. Grade

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the following SQL script:

```
INSERT ALL
INTO Grade (Grade, Status) VALUES ('A+', 'Pass')
INTO Grade (Grade, Status) VALUES ('A', 'Pass')
INTO Grade (Grade, Status) VALUES ('B+', 'Pass')
INTO Grade (Grade, Status) VALUES ('B', 'Pass')
INTO Grade (Grade, Status) VALUES ('C+', 'Pass')
INTO Grade (Grade, Status) VALUES ('C', 'Pass')
INTO Grade (Grade, Status) VALUES ('D+', 'Pass')
INTO Grade (Grade, Status) VALUES ('D', 'Pass')
INTO Grade (Grade, Status) VALUES ('E+', 'Fail')
INTO Grade (Grade, Status) VALUES ('E', 'Fail')
INTO Grade (Grade, Status) VALUES ('F', 'Fail')
SELECT * FROM dual;
COMMIT;
```

The 'Script Output' window below the worksheet shows the results of the execution:

```
11 rows inserted.

Commit complete.
```

Figure 17: INSERT Grade

9.6. Department

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'Coursework' under 'Oracle Connections' and various tables like ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, and TEACHER. Below this is the 'Reports' section with options like All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports.

The main workspace contains a 'Worksheet' tab with the following SQL code:

```
INSERT ALL
  INTO Department (department_name) VALUES ('RTE')
  INTO Department (department_name) VALUES ('Finance')
  INTO Department (department_name) VALUES ('Student Services')
  INTO Department (department_name) VALUES ('External Partnership')
  INTO Department (department_name) VALUES ('Student Career')
  SELECT * FROM dual;
COMMIT;
```

Below the worksheet is a 'Script Output' window showing the results of the execution:

```
Task completed in 0.039 seconds
5 rows inserted.
Commit complete.
```

Figure 18: INSERT Department

9.7. Address

The screenshot shows the Oracle SQL Developer interface. The top window is titled 'Worksheet' and contains a script named 'Coursework~1'. The script is an 'INSERT ALL' statement into the 'ADDRESS' table, inserting 12 rows with various street names, numbers, and states. The bottom window is titled 'Script Output' and shows the results of the execution: '12 rows inserted.' and 'Commit complete.'

```
Worksheet | Query Builder
SELECT * FROM dual;
COMMIT; |
```

```
Script Output | Task completed in 0.04 seconds
12 rows inserted.
Commit complete.
```

Figure 19: INSERT Address

9.8. Assignment

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar on the left lists 'Coursework' under 'Oracle Connections'. The 'Tables (filtered)' section within 'Coursework' contains tables such as ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, and TEACHER. The 'Reports' sidebar lists various report types like Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports.

The main workspace displays a query in the 'Worksheet' tab:

```
INSERT ALL
  INTO Assignment (Assignment_Type) VALUES ('Coursework')
  INTO Assignment (Assignment_Type) VALUES ('Written Exam')
  INTO Assignment (Assignment_Type) VALUES ('Individual Assignment')
  INTO Assignment (Assignment_Type) VALUES ('Group Assignment')
  INTO Assignment (Assignment_Type) VALUES ('Unseen Exam')
SELECT * FROM dual;
COMMIT;
```

The 'Script Output' tab at the bottom shows the results of the execution:

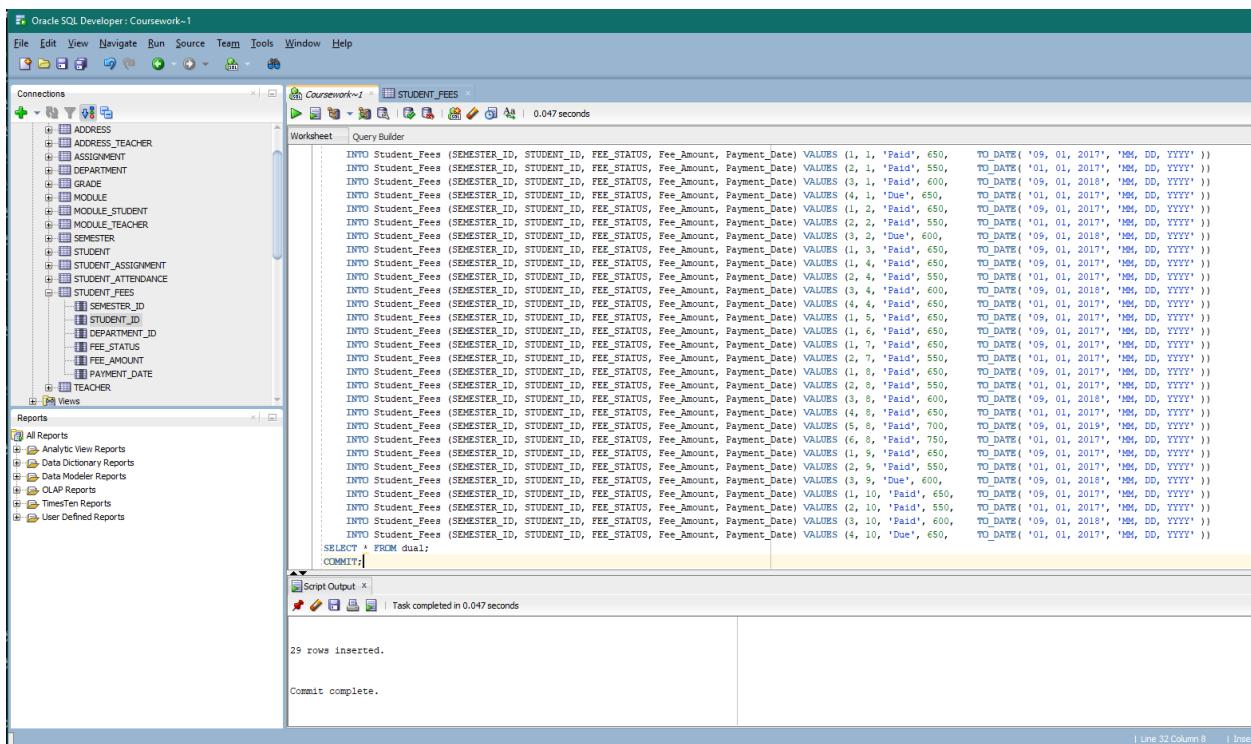
```
Task completed in 0.03 seconds

5 rows inserted.

Commit complete.
```

Figure 20: INSERT Assignment

9.9. Student_Fees



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** A tree view showing various database objects like ADDRESS, ASSIGNMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, and STUDENT_FEES.
- Worksheet:** The main area where the SQL query is entered. The query inserts 29 rows into the STUDENT_FEES table. The columns are SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, and Payment_Date. The Payment_Date values are generated using the TO_DATE function with specific date strings.
- Script Output:** A panel at the bottom showing the results of the query execution. It displays "29 rows inserted." and "Commit complete."
- Toolbar:** Standard SQL Developer toolbar with icons for file operations, navigation, and tools.
- Status Bar:** Shows "Task completed in 0.047 seconds".

```

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 1, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 1, 'Paid', 550, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 1, 'Paid', 600, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (4, 1, 'Due', 650, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 2, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 2, 'Paid', 550, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 2, 'Due', 600, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 3, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 3, 'Paid', 550, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 3, 'Paid', 600, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (4, 3, 'Paid', 650, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 4, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 4, 'Paid', 550, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 4, 'Paid', 600, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (4, 4, 'Paid', 650, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 5, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 5, 'Paid', 550, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 5, 'Paid', 600, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (4, 5, 'Paid', 650, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 6, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 7, 'Paid', 650, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 7, 'Paid', 550, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 8, 'Paid', 650, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 8, 'Paid', 550, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 8, 'Paid', 600, TO_DATE('01, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (4, 8, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (5, 8, 'Paid', 700, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 9, 'Paid', 650, TO_DATE('09, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 9, 'Paid', 550, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 9, 'Due', 600, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (1, 10, 'Paid', 650, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (2, 10, 'Paid', 550, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (3, 10, 'Paid', 600, TO_DATE('01, 01, 2017', 'MM, DD, YYYY'))  

INTO Student_Fees (SEMESTER_ID, STUDENT_ID, FEE_STATUS, Fee_Amount, Payment_Date) VALUES (4, 10, 'Due', 650, TO_DATE('09, 01, 2018', 'MM, DD, YYYY'))  

SELECT * FROM dual;  

COMMIT;

```

Figure 21: INSERT Student_Fees

9.10. Student_Attendance

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database structure under 'Coursework' connection, including tables like ADDRESS, STUDENT, and ATTENDANCE. The central 'Worksheet' tab contains an 'INSERT ALL' statement for the 'Student_Attendance' table. The 'Script Output' tab at the bottom shows the results of the execution.

```

INSERT ALL
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (1, 1, 90)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (1, 2, 98)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (1, 3, 100)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (2, 1, 100)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (2, 2, 81)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (3, 1, 84)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (4, 1, 88)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (4, 2, 89)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (4, 3, 99)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (4, 4, 80)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (5, 1, 80)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (6, 1, 81)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (7, 1, 81)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (7, 2, 89)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (8, 1, 98)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (8, 2, 82)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (8, 3, 89)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (8, 4, 89)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (8, 5, 99)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (8, 6, 100)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (9, 1, 100)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (9, 2, 100)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (10, 1, 100)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (10, 2, 100)
INTO Student_Attendance (Student_ID, Semester_ID, Attendance_Percentage) VALUES (10, 3, 90)
SELECT * FROM dual;
COMMIT;

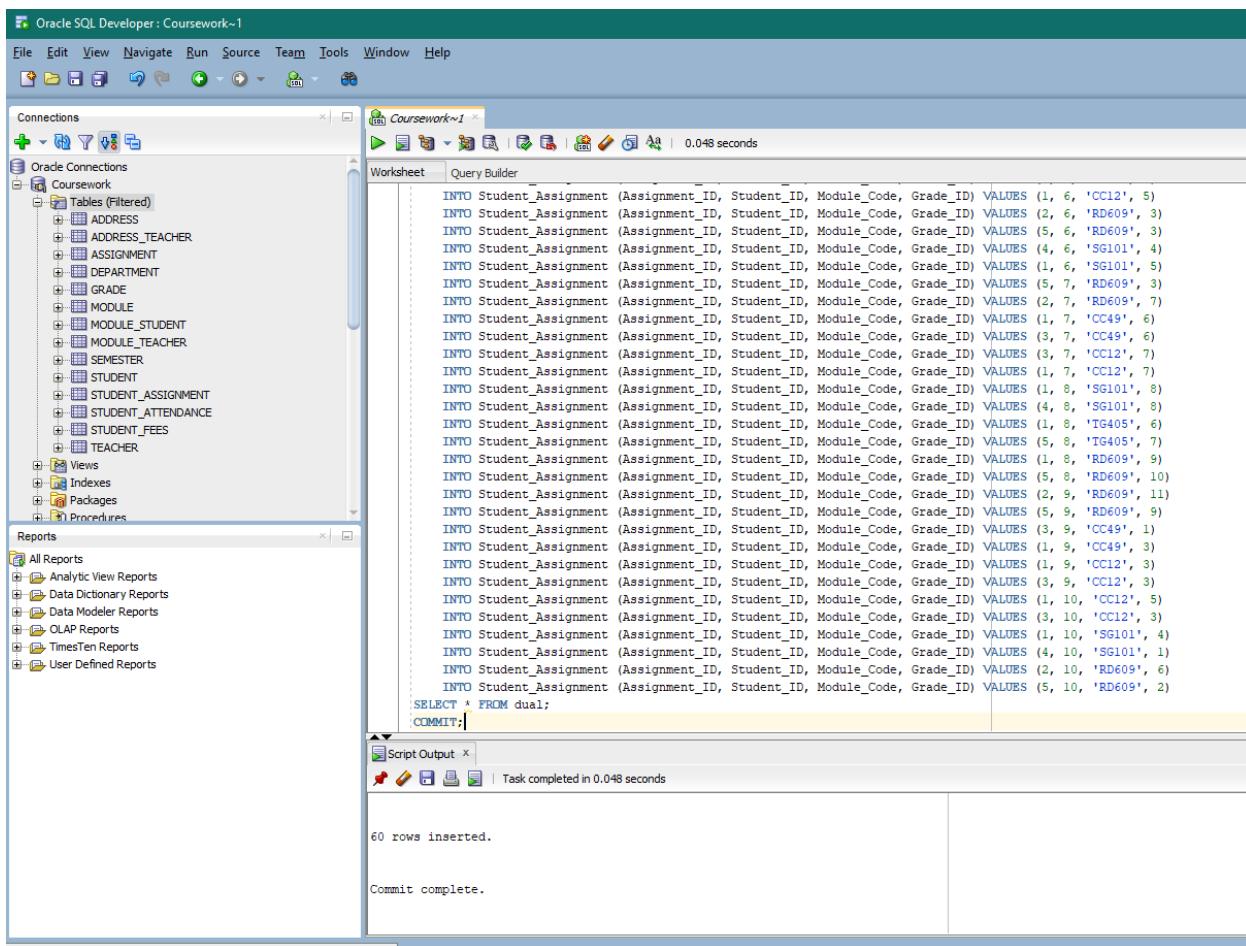
```

Script Output x | Task completed in 0.04 seconds

25 rows inserted.
Commit complete.

Figure 22: INSERT Student_Attendance

9.11. Student_Assignment



The screenshot shows the Oracle SQL Developer interface with the following details:

- File Menu:** File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help.
- Connections:** Oracle Connections, Coursework, Tables (Filtered), ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, TEACHER, Views, Indexes, Packages, Procedures.
- Reports:** All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, User Defined Reports.
- Worksheet:** The main area contains the following SQL code:

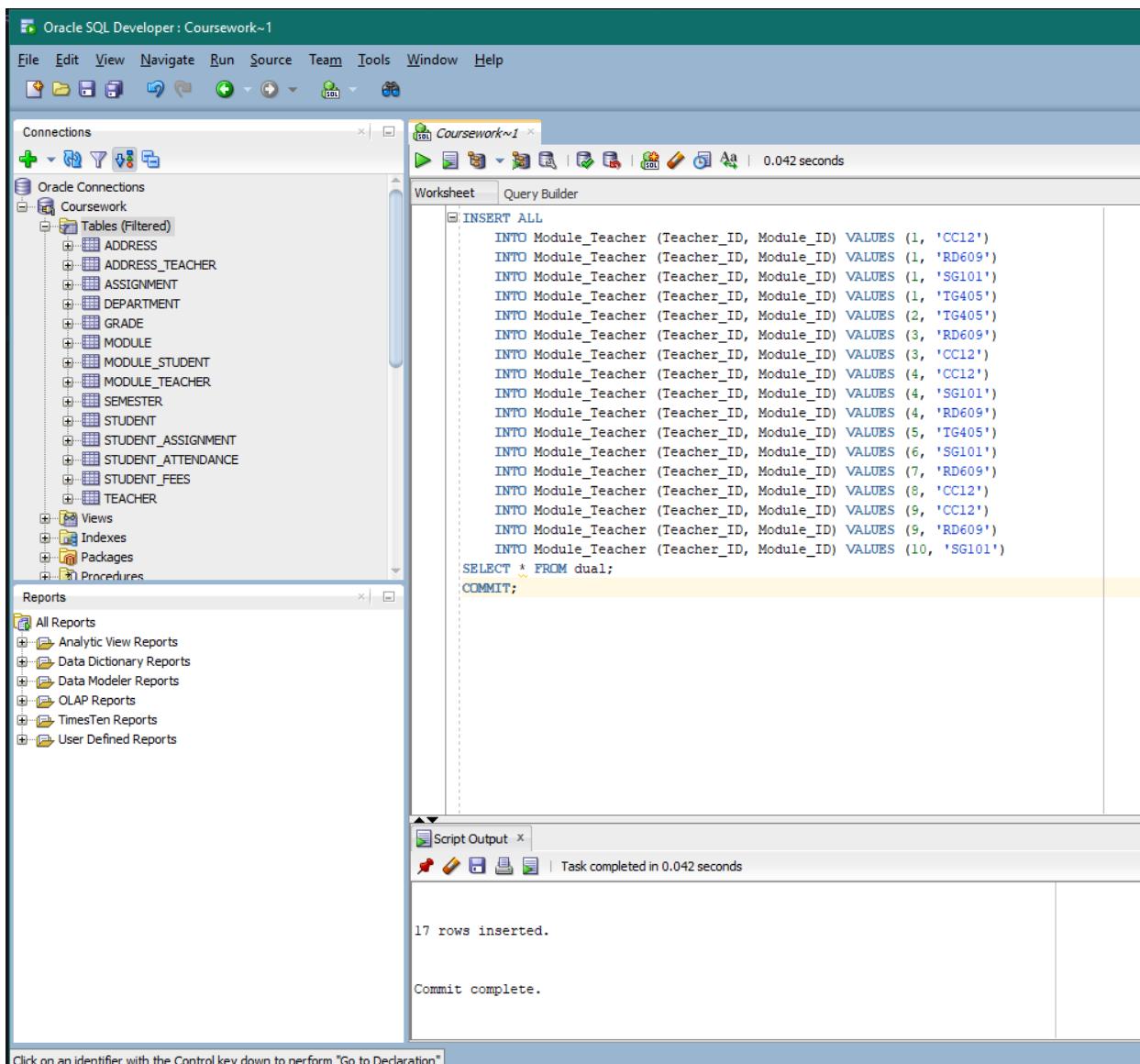
```

INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 6, 'CC12', 5)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (2, 6, 'RD609', 3)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (5, 6, 'RD609', 3)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (4, 6, 'SG101', 4)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 6, 'SG101', 5)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (5, 7, 'RD609', 3)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (2, 7, 'RD609', 7)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 7, 'CC49', 6)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (3, 7, 'CC49', 6)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (3, 7, 'CC12', 7)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 7, 'CC12', 7)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 8, 'SG101', 8)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (4, 8, 'SG101', 8)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 8, 'TG405', 6)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (5, 8, 'TG405', 7)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 8, 'RD609', 9)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (5, 8, 'RD609', 10)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (2, 9, 'RD609', 11)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (5, 9, 'RD609', 9)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (3, 9, 'CC49', 1)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 9, 'CC49', 3)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (3, 9, 'CC12', 3)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 10, 'CC12', 5)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (3, 10, 'CC12', 3)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (1, 10, 'SG101', 4)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (4, 10, 'SG101', 1)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (2, 10, 'RD609', 6)
INTO Student_Assignment (Assignment_ID, Student_ID, Module_Code, Grade_ID) VALUES (5, 10, 'RD609', 2)
SELECT * FROM dual;
COMMIT;

```
- Script Output:** The output shows "60 rows inserted." and "Commit complete.".

Figure 23: INSERT Student_Assignment

9.12. Module_Teacher



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which is expanded to show the 'Coursework' connection, its tables (e.g., ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, TEACHER), and various reports like Analytic View Reports, Data Dictionary Reports, etc. The main workspace contains a 'Worksheet' tab with the following SQL script:

```

INSERT ALL
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (1, 'CC12')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (1, 'RD609')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (1, 'SG101')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (1, 'TG405')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (2, 'TG405')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (3, 'RD609')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (3, 'CC12')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (4, 'CC12')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (4, 'SG101')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (4, 'RD609')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (5, 'TG405')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (6, 'SG101')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (7, 'RD609')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (8, 'CC12')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (9, 'CC12')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (9, 'RD609')
  INTO Module_Teacher (Teacher_ID, Module_ID) VALUES (10, 'SG101')
SELECT * FROM dual;
COMMIT;

```

The 'Script Output' tab at the bottom shows the results of the execution:

```

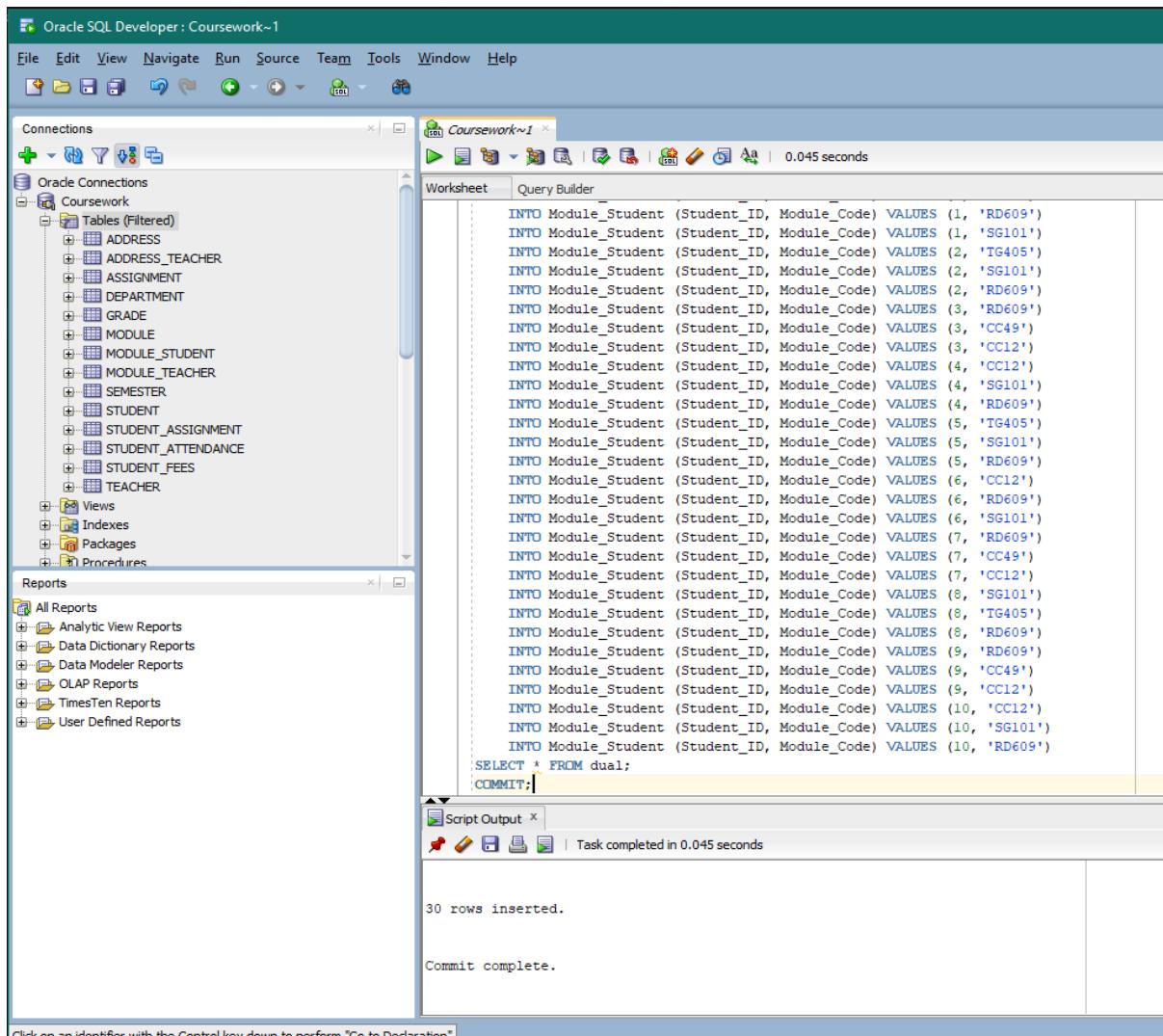
17 rows inserted.

Commit complete.

```

Figure 24: INSERT Module_Teacher

9.13. Module_Student



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** A tree view under "Coursework" shows tables like ADDRESS, ADDRESS_TEACHER, ASSIGNMENT, DEPARTMENT, GRADE, MODULE, MODULE_STUDENT, MODULE_TEACHER, SEMESTER, STUDENT, STUDENT_ASSIGNMENT, STUDENT_ATTENDANCE, STUDENT_FEES, and TEACHER.
- Worksheet:** The main workspace contains the following SQL code:

```

INTO Module_Student (Student_ID, Module_Code) VALUES (1, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (1, 'SG101')
INTO Module_Student (Student_ID, Module_Code) VALUES (2, 'TG405')
INTO Module_Student (Student_ID, Module_Code) VALUES (2, 'SG101')
INTO Module_Student (Student_ID, Module_Code) VALUES (2, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (3, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (3, 'CC49')
INTO Module_Student (Student_ID, Module_Code) VALUES (3, 'CC12')
INTO Module_Student (Student_ID, Module_Code) VALUES (4, 'CC12')
INTO Module_Student (Student_ID, Module_Code) VALUES (4, 'SG101')
INTO Module_Student (Student_ID, Module_Code) VALUES (4, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (5, 'TG405')
INTO Module_Student (Student_ID, Module_Code) VALUES (5, 'SG101')
INTO Module_Student (Student_ID, Module_Code) VALUES (5, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (6, 'CC12')
INTO Module_Student (Student_ID, Module_Code) VALUES (6, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (6, 'SG101')
INTO Module_Student (Student_ID, Module_Code) VALUES (7, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (7, 'CC49')
INTO Module_Student (Student_ID, Module_Code) VALUES (7, 'CC12')
INTO Module_Student (Student_ID, Module_Code) VALUES (8, 'SG101')
INTO Module_Student (Student_ID, Module_Code) VALUES (8, 'TG405')
INTO Module_Student (Student_ID, Module_Code) VALUES (8, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (9, 'RD609')
INTO Module_Student (Student_ID, Module_Code) VALUES (9, 'CC49')
INTO Module_Student (Student_ID, Module_Code) VALUES (9, 'CC12')
INTO Module_Student (Student_ID, Module_Code) VALUES (10, 'CC12')
INTO Module_Student (Student_ID, Module_Code) VALUES (10, 'SG101')
INTO Module_Student (Student_ID, Module_Code) VALUES (10, 'RD609')

SELECT * FROM dual;
COMMIT;

```
- Script Output:** The output pane shows the results of the execution:

```

30 rows inserted.

Commit complete.

```

Figure 25: INSERT Module_Student

9.14. Address_Teacher

The screenshot shows the Oracle SQL Developer interface. The Worksheet pane contains an SQL script for inserting data into the Address_Teacher table:

```
INSERT ALL
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (1, 2)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (2, 1)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (2, 3)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (3, 4)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (3, 5)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (4, 6)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (5, 7)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (6, 8)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (7, 9)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (8, 10)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (9, 11)
INTO Address_Teacher (Teacher_ID, Address_ID) VALUES (10, 12)
SELECT * FROM dual;
COMMIT;
```

The Script Output pane shows the results of the execution:

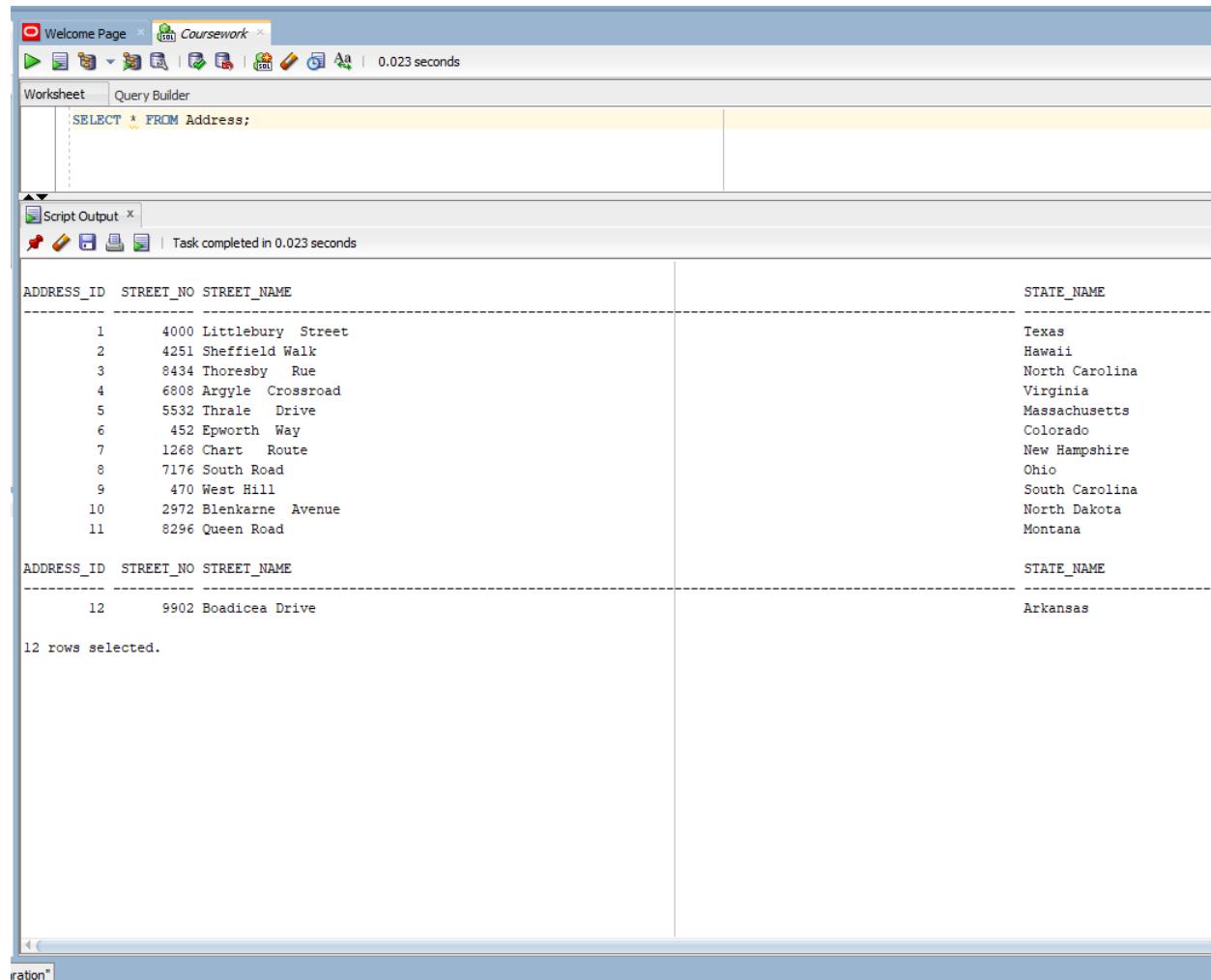
```
12 rows inserted.

Commit complete.
```

Figure 26: INSERT Address_Teacher

10. SELECT Statements

10.1. Address



The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page' and 'Coursework'. Below the tabs, the 'Worksheet' tab is selected, and the 'Query Builder' icon is visible. The main area displays a query in the worksheet:

```
SELECT * FROM Address;
```

Below the worksheet, the 'Script Output' tab is active, showing the results of the query. The output is presented in two parts. The first part shows 11 rows of address data with their corresponding state names:

ADDRESS_ID	STREET_NO	STREET_NAME	STATE_NAME
1	4000	Littlebury Street	Texas
2	4251	Sheffield Walk	Hawaii
3	8434	Thoresby Rue	North Carolina
4	6808	Argyle Crossroad	Virginia
5	5532	Thrale Drive	Massachusetts
6	452	Epworth Way	Colorado
7	1268	Chart Route	New Hampshire
8	7176	South Road	Ohio
9	470	West Hill	South Carolina
10	2972	Blenkarne Avenue	North Dakota
11	8296	Queen Road	Montana

The second part of the output shows one additional row:

ADDRESS_ID	STREET_NO	STREET_NAME	STATE_NAME
12	9902	Boadicea Drive	Arkansas

At the bottom of the output, it says '12 rows selected.'

Figure 27:Select Address

10.2. Address_Teacher

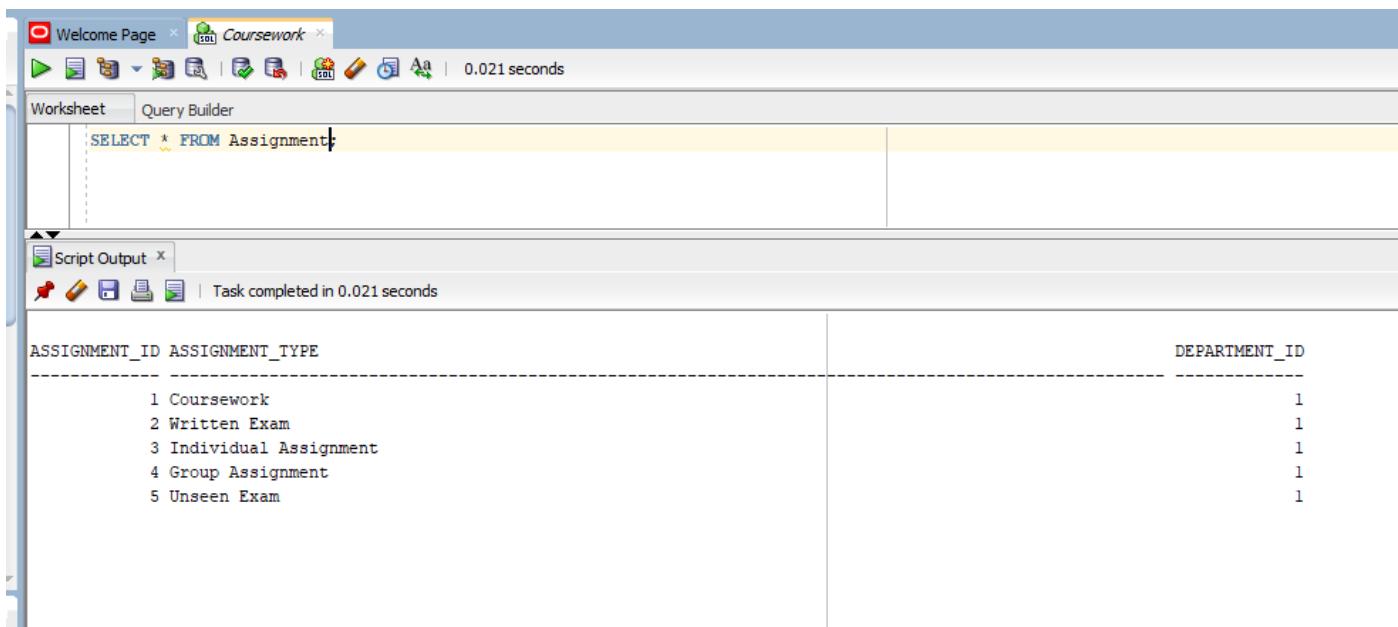
The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the SQL query: `SELECT * FROM Address_Teacher;`. Below the query, the results are shown in two parts. The first part displays rows 1 through 11, and the second part displays row 12. The output is as follows:

ADDRESS_ID	TEACHER_ID
1	2
2	1
3	2
4	3
5	3
6	4
7	5
8	6
9	7
10	8
11	9
12	10

12 rows selected.

Figure 28:Select Address_Teacher

10.3. Assignment



The screenshot shows a SQL Server Management Studio (SSMS) interface. In the top bar, there are tabs for 'Welcome Page' and 'Coursework'. Below the tabs is a toolbar with various icons. The main area has two tabs: 'Worksheet' and 'Query Builder', with 'Worksheet' selected. The worksheet contains the following SQL query:

```
SELECT * FROM Assignment;
```

Below the worksheet is a 'Script Output' window showing the results of the query. The output is a table with three columns: 'ASSIGNMENT_ID', 'ASSIGNMENT_TYPE', and 'DEPARTMENT_ID'. The data is as follows:

ASSIGNMENT_ID	ASSIGNMENT_TYPE	DEPARTMENT_ID
1	Coursework	1
2	Written Exam	1
3	Individual Assignment	1
4	Group Assignment	1
5	Unseen Exam	1

Figure 29:Select Assignment

10.4. Grade

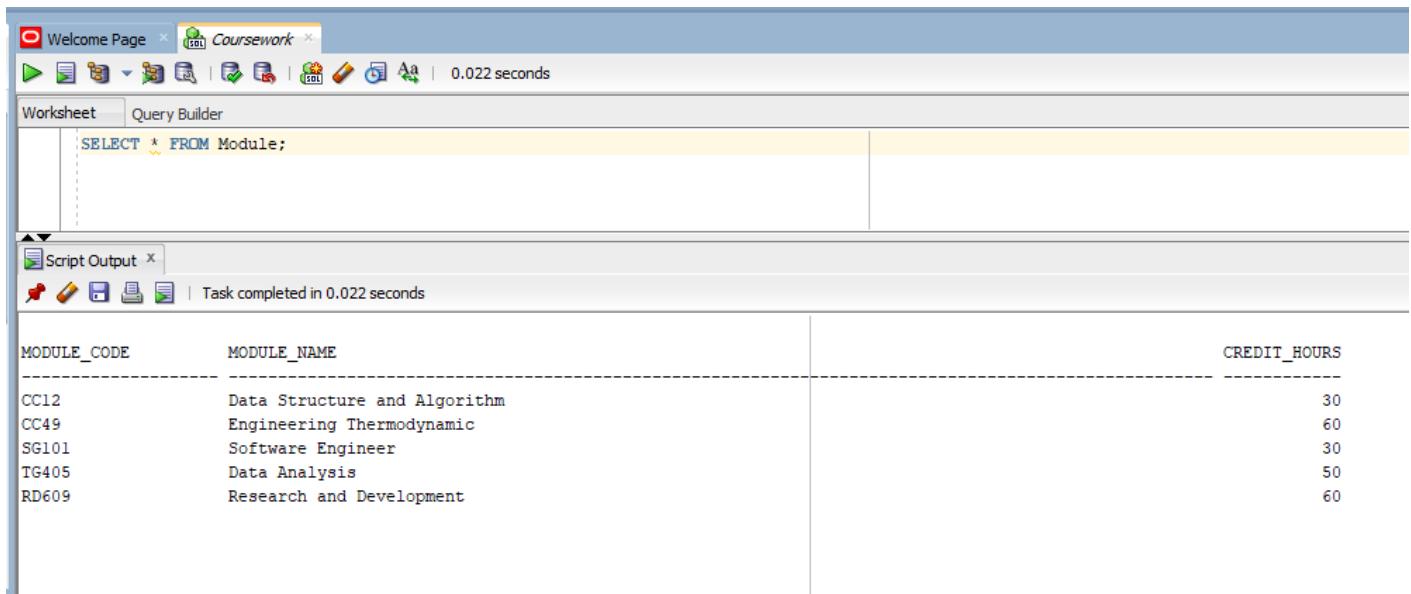
The screenshot shows a database management interface with a toolbar at the top containing various icons for file operations, queries, and preferences. The main area is divided into two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying the SQL query: 'SELECT * FROM Grade;'. Below the query, the results are shown in a table format:

GRADE_ID	GR	STATUS
1	A+	Pass
2	A	Pass
3	B+	Pass
4	B	Pass
5	C+	Pass
6	C	Pass
7	D+	Pass
8	D	Pass
9	E+	Fail
10	E	Fail
11	F	Fail

Below the table, a message indicates '11 rows selected.'

Figure 30:Select Grade

10.5. Module



The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page' and 'Coursework'. Below the bar, there are various icons for database management and a status message '0.022 seconds'. The main area has two tabs: 'Worksheet' and 'Query Builder', with 'Worksheet' selected. The worksheet contains the SQL query: 'SELECT * FROM Module;'. To the right of the query, the results are displayed in a table format:

MODULE_CODE	MODULE_NAME	CREDIT_HOURS
CC12	Data Structure and Algorithm	30
CC49	Engineering Thermodynamic	60
SG101	Software Engineer	30
TG405	Data Analysis	50
RD609	Research and Development	60

Below the table, a message indicates 'Task completed in 0.022 seconds'. The bottom of the interface shows a toolbar with various icons.

Figure 31:Select Module

10.6. Module_Student

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the SQL query:

```
SELECT * FROM Module_Student;
```

The 'Script Output' tab shows the results of the query, which are repeated three times. Each result set has a header:

STUDENT_ID	MODULE_CODE
2	TG405
3	CC12
3	CC49
3	RD609
4	CC12
4	RD609

STUDENT_ID	MODULE_CODE
4	SG101
5	RD609
5	SG101
5	TG405
6	CC12
6	RD609
6	SG101
7	CC12
7	CC49
7	RD609
8	RD609

STUDENT_ID	MODULE_CODE
8	SG101
8	TG405
9	CC12
9	CC49
9	RD609
10	CC12
10	RD609
10	SG101

Below the tables, the message '30 rows selected.' is displayed.

Figure 32:Select Module_Student

10.1. Module_Teacher

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page' and 'Coursework'. Below the tabs is a toolbar with various icons for database operations like insert, update, delete, and search. The main area has two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active and contains the SQL query: 'SELECT * FROM Module_Teacher;'. Below the query, the results are displayed in two parts. The first part shows data for modules CC12, RD609, and SG101, with teacher IDs 1, 3, 4, 8, 9, 1, 3, 4, 7, 9, and 1 respectively. The second part shows data for module TG405, with teacher IDs 1, 2, and 5. A message at the bottom of the results area states 'Task completed in 0.016 seconds'. At the very bottom of the interface, it says '17 rows selected.'

MODULE_ID	TEACHER_ID
CC12	1
CC12	3
CC12	4
CC12	8
CC12	9
RD609	1
RD609	3
RD609	4
RD609	7
RD609	9
SG101	1
MODULE_ID	TEACHER_ID
SG101	4
SG101	6
SG101	10
TG405	1
TG405	2
TG405	5

Figure 33:Select Module_Teacher

10.1. Semester

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page' and 'Coursework'. Below the tabs is a toolbar with various icons. The main area has two tabs: 'Worksheet' and 'Query Builder', with 'Worksheet' selected. A code editor window contains the SQL query:

```
SELECT * FROM Semester;
```

Below the code editor is a 'Script Output' window showing the results of the query. The output header is:

SEMESTER_ID	SEMESTER_FEE	SEMESTER
1	650	1
2	550	2
3	600	3
4	650	4
5	700	5
6	750	6

The results show six rows of data. At the bottom of the output window, it says '6 rows selected.'

Figure 34:Select Semester

10.1. Student

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the SQL query: `SELECT * FROM Student;`. Below the worksheet, the 'Script Output' tab shows the results of the query execution. The output includes the student ID, name, and street number, with a total of 10 rows selected.

STUDENT_ID	STUDENT_NAME	STREET_NO
1	Michelle Grady	6342
2	Sara Anderson	9896
3	Cherish Leigh	8296
4	Sebastian Ballard	9902
5	Sydney London	5238
6	Hannah Giles	7889
7	Jack Hammond	9827
8	Phillip Morrow	7410
9	Carissa Clarke	8580
10	Deborah Bingham	3771

10 rows selected.

Figure 35:Select Student

10.1. Student_Assignment

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page' and 'Coursework'. Below the tabs, there are several icons for navigating between windows and performing database operations. The main area has two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying the SQL query:

```
SELECT * FROM Student_Assignment;
```

Below the query, the 'Script Output' window displays the results of the query execution. It shows three distinct sets of data, each starting with a header row and followed by multiple data rows. The first set of data is:

ASSIGNMENT_ID	STUDENT_ID	MODULE_CODE	GRADE_ID
1	6	SG101	5
5	7	RD609	3
2	7	RD609	7
1	7	CC49	6
3	7	CC49	6
3	7	CC12	7
1	7	CC12	7
1	8	SG101	8
4	8	SG101	8

The second set of data is:

ASSIGNMENT_ID	STUDENT_ID	MODULE_CODE	GRADE_ID
1	8	TG405	6
5	8	TG405	7
1	8	RD609	9
5	8	RD609	10
2	9	RD609	11
5	9	RD609	9
3	9	CC49	1
1	9	CC49	3
1	9	CC12	3
3	9	CC12	3
1	10	CC12	5

The third set of data is:

ASSIGNMENT_ID	STUDENT_ID	MODULE_CODE	GRADE_ID
3	10	CC12	3
1	10	SG101	4
4	10	SG101	1
2	10	RD609	6
5	10	RD609	2

At the bottom of the output window, the message '60 rows selected.' is displayed.

Figure 36:Select Student_Assignment

10.1. Student_Attendance

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page' and 'Coursework'. Below the tabs is a toolbar with various icons for database operations like insert, update, delete, and search. The main area has two tabs: 'Worksheet' and 'Query Builder', with 'Worksheet' selected. A query is entered in the worksheet:

```
SELECT * FROM Student_Attendance;
```

Below the worksheet is a 'Script Output' window. It displays the results of the query execution. The output is divided into three sections by dashed horizontal lines:

STUDENT_ID	SEMESTER_ID	ATTENDANCE_PERCENTAGE	DEPARTMENT_ID
1	1	90	1
1	2	98	1
1	3	100	1
2	1	100	1
2	2	81	1
3	1	84	1
4	1	88	1
4	2	89	1
4	3	99	1
4	4	80	1
5	1	80	1

STUDENT_ID	SEMESTER_ID	ATTENDANCE_PERCENTAGE	DEPARTMENT_ID
6	1	81	1
7	1	81	1
7	2	89	1
8	1	98	1
8	2	82	1
8	3	89	1
8	4	89	1
8	5	99	1
8	6	100	1
9	1	100	1
9	2	100	1

STUDENT_ID	SEMESTER_ID	ATTENDANCE_PERCENTAGE	DEPARTMENT_ID
10	1	100	1
10	2	100	1
10	3	100	1

At the bottom of the script output window, it says '25 rows selected.'

Figure 37:Select Attendance

10.1. Student_Fees

The screenshot shows the Oracle SQL Developer interface. On the left, the Object Navigator displays various database tables under the 'STUDENT_FEES' schema. In the center, the Worksheet pane contains the SQL query: 'SELECT * FROM STUDENT_FEES;'. Below the query, the Script Output pane shows the results of the execution.

SEMESTER_ID	STUDENT_ID	DEPARTMENT_ID	FEES_STATUS	FEES_AMOUNT	PAYMENT_DATE
2	8	2	Paid	550	01-JAN-17
3	8	2	Paid	600	01-SEP-18
4	8	2	Paid	650	01-JAN-17
5	8	2	Paid	700	01-SEP-19
6	8	2	Paid	750	01-JAN-17
1	9	2	Paid	650	01-SEP-17
2	9	2	Paid	550	01-JAN-17
3	9	2	Due	600	01-SEP-18
1	10	2	Paid	650	01-SEP-17
2	10	2	Paid	550	01-JAN-17
3	10	2	Paid	600	01-SEP-18
4	10	2	Due	650	01-JAN-17

29 rows selected.

Figure 38:Select Fees

10.1. Teacher

The screenshot shows a database interface with a query window and a script output window.

Query Window:

```
SELECT * FROM Teacher;
```

Script Output:

Task completed in 0.365 seconds

TEACHER_ID	TEACHER_NAME	EMAIL
1	Ethan Cunningham	Ethan_Cunningham2529@bretv.org
2	Tyson McCall	Tyson_Mccall1386@deons.tech
3	Alexander Lee	Alexander_Lee2f63@vetan.org
4	Gil Garner	Gil_Garner873@twipet.com
5	Carmen Andrews	Carmen_Andrews586@typill.biz
6	Peter Graham	Peter_Graham5480@bulaffy.com
7	Anthony Gates	Anthony_Gates1413@quentu.biz
8	Carl Alexander	Carl_Alexander3633@cispeto.com
9	Alessia Underhill	Alessia_Underhill12045@gmail.com
10	Kimberly Payne	Kimberly_Payne1227@iatim.tech

10 rows selected.

Figure 39:Select Teacher

11. Forms

11.1. Dashboard

The screenshot shows the Berkeley College Management System dashboard. At the top, there is a navigation bar with a red 'B' logo and links for Teacher, Address, Module, Department, Student, and Teacher-Module, Student-Fees, Student-Assignment. Below the navigation bar, the title 'Berkeley College Management System' is displayed. A section titled 'Available webforms:' contains eight cards arranged in two rows of four. The top row consists of five 'Basic Form' cards: 'Teacher Details' (red), 'Student Details' (red), 'Module Details' (red), 'Address Details' (red), and 'Departemnt Details' (red). The bottom row consists of three 'Complex Form' cards: 'Teacher-Module Details' (teal), 'Student-Fees Details' (teal), and 'Student_Assignment Details' (teal).

Form Type	Form Name	Description
Basic Form	Teacher Details	View and edit teacher details through this form.
Basic Form	Student Details	View, add, update or delete student details through this form.
Basic Form	Module Details	View, add, update or delete module details through this form.
Basic Form	Address Details	View, add, update or delete address details through this form.
Basic Form	Departemnt Details	View, add, update or delete department details through this form.
Complex Form	Teacher-Module Details	View teacher details including their associated module details through this form.
Complex Form	Student-Fees Details	View student details along with their fee details through this form.
Complex Form	Student_Assignment Details	View student details along with their assignment details through this form.

Figure 40: WebForms Dashboard

11.2. Basic Forms

11.2.1. Teacher Form

The screenshot shows a web-based application interface for managing teacher data. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there are input fields for Name and Email, both with placeholder text and small circular icons. A dropdown menu for Student_ID is set to Null. A blue 'Add' button is located below these fields. The main area displays a table with 10 rows of teacher data, each with Edit and Delete links. The columns are labeled ID, Name, EMAIL, and Student ID.

	ID	Name	EMAIL	Student ID
Edit	1	Ethan Cunningham	Ethan_Cunningham2529@breyt.org	
Edit	2	Tyson McCall	Tyson_Mccall3866@deons.tech	
Edit	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit	10	Kimberly Payne	Kimberly_Payne1227@latim.tech	

Figure 41: Teacher Form

11.2.2. Address Form

The screenshot shows a web-based application interface for managing addresses. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there are three input fields: 'Street Number' (with value 0), 'Street Name' (with value 0), and 'State Name' (with value 0). A blue 'Add' button is located below these fields. Below the input fields is a table listing 11 address records:

	ID	Street Number	Street Name	State Name
Edit	1	4000	Littlebury Street	Texas
Edit	2	4251	Sheffield Walk	Hawaii
Edit	3	8434	Thoresby Rue	North Carolina
Edit	4	6808	Argyle Crossroad	Virginia
Edit	5	5532	Thrall Drive	Massachusetts
Edit	6	452	Epworth Way	Colorado
Edit	7	1268	Chart Route	New Hampshire
Edit	8	7176	South Road	Ohio
Edit	9	470	West Hill	South Carolina
Edit	10	2972	Blenkarne Avenue	North Dakota
Edit	11	8296	Queen Road	Montana

Figure 42: Teacher Form

11.2.3. Module Form

The screenshot shows a web-based application for managing modules. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there are three input fields: 'Module Code' (with value CC12), 'Name' (with value Data Structure and Algorithm), and 'Credit Hours' (with value 30). A blue 'Add' button is located below these fields. Below the input fields is a table listing five module records:

	Module Code	Name	Credit Hours
Edit Delete	CC12	Data Structure and Algorithm	30
Edit Delete	CC49	Engineering Thermodynamic	60
Edit Delete	SG101	Software Engineer	30
Edit Delete	TG405	Data Analysis	50
Edit Delete	RD609	Research and Development	60

Figure 43: Module Form

11.2.4. Department Form

The screenshot shows a web-based application interface for managing departments. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, and Teacher-Module, Student-Fees, Student-Assignment. Below the navigation bar, there is a search input field labeled "Department" with a placeholder "(empty)" and a red clear icon. A blue "Add" button is located below the search field. The main content area displays a table with the following data:

	ID	Department
Edit Delete	1	RTE
Edit Delete	2	Finance
Edit Delete	3	Student Services
Edit Delete	4	External Partnership
Edit Delete	5	Student Career

Figure 44: Department Form

11.2.5. Student Form

The screenshot shows a web-based application for managing student addresses. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there are four input fields labeled 'Name', 'Street Number', 'Street Name', and 'State Name', each with a small red circular icon containing a question mark. Below these fields is a blue 'Add' button. Underneath the 'Add' button is a table with ten rows of data, each representing a student's address. The table has columns for ID, Name, Street Number, Street Name, and State Name. Each row includes 'Edit' and 'Delete' links in the first column. The data is as follows:

	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Deborah Bingham	3771	Chester Grove	North Carolina

Figure 45: Student Form

11.3. Complex Forms

11.3.1. Teacher-Module Mapping

11.3.1.1. Form

B Teacher Address Module Department Student | Teacher-Module Student-Fees Student-Assignment

Table of Teachers				
	ID	Name	EMAIL	Student ID
Select	1	Ethan Cunningham	Ethan_Cunningham2529@bretv.org	
Select	2	Tyson McCall	Tyson_Mccall3866@deons.tech	
Select	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Select	4	Gil Garner	Gil_Garner8738@twipet.com	
Select	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Select	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Select	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Select	8	Carl Alexander	Carl_Alexander3633@cspeto.com	
Select	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Select	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	

Table of Associated Modules

Please select a teacher teaching atleast one module to view their associated modules.

Figure 46: Teacher Module Mapping Form Before Selection

The screenshot shows a database interface with two tables. The top part displays the 'Table of Teachers' with columns: ID, Name, EMAIL, and Student ID. The bottom part displays the 'Table of Associated Modules' with columns: Module Code, Teacher ID, Module Name, and Credit Hours.

Table of Teachers

	ID	Name	EMAIL	Student ID
Select	1	Ethan Cunningham	Ethan_Cunningham2529@breyt.org	
Select	2	Tyson McCall	Tyson_Mccall3866@deons.tech	
Select	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Select	4	Gil Garner	Gil_Garner8738@twipet.com	
Select	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Select	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Select	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Select	8	Carl Alexander	Carl_Alexander3633@clspeto.com	
Select	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Select	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	

Table of Associated Modules

Module Code	Teacher ID	Module Name	Credit Hours
CC12	4	Data Structure and Algorithm	30
RD609	4	Research and Development	60
SG101	4	Software Engineer	30

Figure 47: Teacher Module Mapping Form After Selection

11.3.1.2. Query

Assuming the selected student has student_id 1, the query is as following:

```
1. SELECT m.Module_Code as "Module Code", mt.Teacher_ID "Teacher ID", m.Module_Name as "Module  
   Name", m.Credit_Hours "Credit Hours"  
2.          FROM Module m  
3.          INNER JOIN Module_Teacher mt ON mt.Module_ID = m.Module_Code  
4.          WHERE mt.Teacher_ID = 1  
5.          ;  
6.
```

In the webform the query string is written as:

```
1. $@"SELECT m.Module_Code as ""Module Code"", mt.Teacher_ID ""Teacher ID"", m.Module_Name as  
   ""Module Name"", m.Credit_Hours ""Credit Hours""  
2.          FROM Module m  
3.          INNER JOIN Module_Teacher mt ON mt.Module_ID = m.Module_Code  
4.          WHERE mt.Teacher_ID = {teacherID}  
5.          ;  
6.
```

11.4. Student-Fees Mapping

11.4.1.1. Form

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Fee Details

Please select a student who has been assigned at least one fee invoice.

Figure 48: Student-Fees Mapping Form Before Selection

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Fee Details

Student ID	Semester	Department Name	Fee Amount	Fee Status
2	1	Finance	650	Paid
2	2	Finance	550	Paid
2	3	Finance	600	Due

Figure 49: Student-Fees Mapping Form After Selection

11.4.1.2. Query

Assuming the selected student has student_id 1, the query is as following:

```
1. SELECT sf.Student_ID "Student ID", se.Semester "Semester", d.Department_Name as "Department
   Name", sf.Fee_Amount "Fee Amount", sf.Fee_Status "Fee Status"
2.          FROM Student s
3.          INNER JOIN Student_Fees sf ON sf.Student_ID = s.Student_ID
4.          INNER JOIN Semester se ON se.Semester_ID = sf.Semester_ID
5.          INNER JOIN Department d ON d.Department_ID =
6.             sf.Department_ID
7.          WHERE sf.Student_ID = 1
8.          ;
```

In the webform the query string is written as:

```
1. $@"SELECT sf.Student_ID ""Student ID"", se.Semester ""Semester"", d.Department_Name as
   ""Department Name"", sf.Fee_Amount ""Fee Amount"", sf.Fee_Status ""Fee Status""
2.          FROM Student s
3.          INNER JOIN Student_Fees sf ON sf.Student_ID = s.Student_ID
4.          INNER JOIN Semester se ON se.Semester_ID = sf.Semester_ID
5.          INNER JOIN Department d ON d.Department_ID =
6.             sf.Department_ID
7.          WHERE sf.Student_ID = {studentID}
8.          ";
```

11.5. Student-Assignment Mapping

11.5.1.1. Form

B Teacher Address Module Department Student | Teacher-Module Student-Fees Student-Assignment

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Assignment Details

Please select a student who has been graded on atleast one assignment.

Figure 50: Student-Assignment Mapping Form Before Selection

B Teacher Address Module Department Student | Teacher-Module Student-Fees Student-Assignment

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Assignment Details

Student ID	Module Code	Module Name	Assignment Type	Department Name	Grade	Status
3	RD609	Research and Development	Written Exam	RTE	A+	Pass
3	RD609	Research and Development	Unseen Exam	RTE	B	Pass
3	CC49	Engineering Thermodynamic	Individual Assignment	RTE	C+	Pass
2	CC49	Engineering Thermodynamic	Coursework	RTE	F+	Fail

Figure 51: Student-Fees Mapping Form After Selection

11.5.1.2. Query

Assuming the selected student has student_id 1, the query is as following:

```

1. SELECT sa.Student_ID "Student ID", m.Module_Code "Module Code" , m.Module_Name "Module
   Name", a.Assignment_Type "Assignment Type" , d.Department_Name "Department Name" , g.Grade
   "Grade", g.Status "Status"
2.                               FROM Student s
3.                               INNER JOIN Student_Assignment sa ON sa.Student_ID =
   s.Student_ID
4.                               INNER JOIN Grade g ON sa.Grade_ID = g.Grade_ID
5.                               INNER JOIN Module m ON sa.Module_Code = m.Module_Code
6.                               INNER JOIN Assignment a ON sa.Assignment_ID =
   a.Assignment_ID
7.                               INNER JOIN Department d ON a.Department_ID = d.Department_ID
8. WHERE sa.Student_ID = 1;
9.
```

In the webform the query string is written as:

```

1. $@"SELECT sa.Student_ID ""Student ID"", m.Module_Code """Module Code"" , m.Module_Name
   """Module Name""", a.Assignment_Type """Assignment Type""",
   d.Department_Name """Department Name"" , g.Grade """Grade""",
   g.Status """Status"""
2.                               FROM Student s
3.                               INNER JOIN Student_Assignment sa ON sa.Student_ID =
   s.Student_ID
4.                               INNER JOIN Grade g ON sa.Grade_ID = g.Grade_ID
5.                               INNER JOIN Module m ON sa.Module_Code = m.Module_Code
6.                               INNER JOIN Assignment a ON sa.Assignment_ID =
   a.Assignment_ID
7.                               INNER JOIN Department d ON a.Department_ID = d.Department_ID
8. WHERE sa.Student_ID = {studentID}
9.                               ";
10.
11.
```

12. User Manual

If the project is run without changing the port settings, then the url for the project is:

<https://localhost:44370/Index.aspx>

12.1. Included Content

1. Landing Page
2. Teacher Form
3. Address Form
4. Module Form
5. Teacher-Module Form
6. Student-Fees Form
7. Student-Assignment Form

12.2. Landing Page

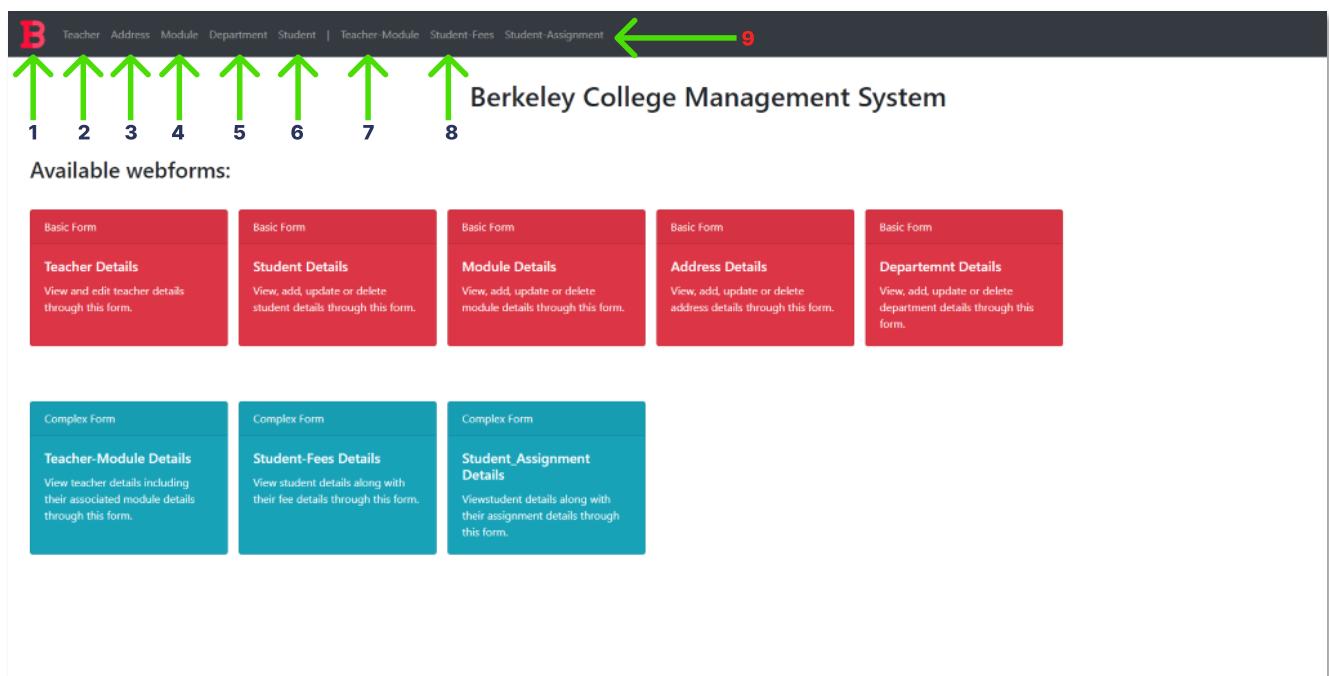


Figure 52: Landing Page

1. The logo can be pressed to navigate to the dashboard
2. The Teacher text in the navbar can be pressed to navigate to the Teacher form
3. The Module text in the navbar can be pressed to navigate to the Module form
4. The Department text in the navbar can be pressed to navigate to the Department form
5. The Student text in the navbar can be pressed to navigate to the Student form
6. The Teacher-Module text in the navbar can be pressed to navigate to the Teacher-Module form
7. The Student-Fees text in the navbar can be pressed to navigate to the Student-Fees form
8. The Student-Assignment text in the navbar can be pressed to navigate to the Student-Assignment form

12.3. Teacher Form

The screenshot shows a web-based application for managing teacher data. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assigment. Below the navigation bar is a form section with three input fields: 'Name' (with a red placeholder 'Required'), 'Email' (with a red placeholder 'Required'), and 'Student ID Null'. An 'Add' button is located below these fields. To the right of the form is a grid table displaying teacher details. The grid has columns for 'ID', 'Name', 'EMAIL', and 'Student ID'. Each row contains an 'Edit' and 'Delete' link followed by the teacher's name and email. The 'Student ID' column shows values like 'Null', '1', '2', etc. Green arrows numbered 1 through 7 point to specific elements: 1 points to the 'Name' field, 2 points to the 'Email' field, 3 points to the 'Student ID Null' dropdown, 4 points to the 'Add' button, 5 points to the 'Student ID' column header, and 6 and 7 point to the 'Edit' and 'Delete' links in the first grid row.

ID	Name	EMAIL	Student ID
Edit Delete 1	Ethan Cunningham	Ethan.Cunningham2529@bretfy.org	
Edit Delete 2	Tyson McCall	Tyson_Mccall3866@deons.tech	
Edit Delete 3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit Delete 4	Gil Garner	Gil_Garner8738@twipet.com	
Edit Delete 5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit Delete 6	Peter Graham	Peter_Graham5489@bulaffy.com	
Edit Delete 7	Anthony Gates	Anthony_Gates1413@guentubiz	
Edit Delete 8	Carl Alexander	Carl_Alexander3633@chspiro.com	
Edit Delete 9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit Delete 10	Kimberly Payne	Kimberly_Payne1227@karlin.tech	

Figure 53: Teacher Form

1. Enter the name of teacher to be added in the name text field. This text field cannot be left empty.
2. Enter the email address of teacher to be added in the email text field. This text field cannot be left empty.
3. If the teacher was a former student, select their student ID from this dropdown. Otherwise, leave it as null.
4. Press the Add button to add data.
5. The teacher details data is displayed in this grid.
6. Press this button to edit data in the grid.
7. Press this button to delete data in the grid.

12.4. Address Form

The screenshot shows a web-based application for managing addresses. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assessment. Below the navigation bar, there are three input fields: 'Street Number' (with a red placeholder 'Street'), 'Street Name' (with a red placeholder 'Name'), and 'State Name' (with a red placeholder 'State'). To the right of each input field is a green arrow labeled 1, 2, and 3 respectively. Below these fields is a blue 'Add' button, which has a green arrow labeled 4 pointing to it. To the left of the 'Add' button, there are two green arrows labeled 6 and 7 pointing upwards, likely indicating edit and delete functions. To the right of the 'Add' button is a green arrow labeled 5 pointing to the left edge of a data grid. The data grid contains 11 rows of address information, each with columns for ID, Street Number, Street Name, and State Name. Each row includes 'Edit' and 'Delete' buttons. The data is as follows:

ID	Street Number	Street Name	State Name
1	4000	Littlebury Street	Texas
2	4251	Sherfield Walk	Henril
3	8434	Thoresby Rue	North Carolina
4	6808	Argyle Crossroad	Virginia
5	5532	Thrale Drive	Massachusetts
6	452	Epworth Way	Colorado
7	1268	Chart Route	New Hampshire
8	7176	South Road	Ohio
9	470	West Hill	South Carolina
10	2972	Blenkarne Avenue	North Dakota
11	8296	Queen Road	Montana

Figure 54: Address Form

1. Enter the Street number of the address in this text field. This text field cannot be left empty.
2. Enter the name of street to be added in this text field. This text field cannot be left empty.
3. Enter the street name of the address to be added in this text field. This text field cannot be left empty.
4. Press the Add button to add data.
5. The teacher details data is displayed in this grid.
6. Press this button to edit data in the grid.
7. Press this button to delete data in the grid.

12.5. Module Form

The screenshot shows a web-based application for managing module details. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assessment. Below the navigation bar, there are three input fields: 'Module Code' (containing 'CC12'), 'Name' (containing 'Data Structure and Algorithm'), and 'Credit Hours' (containing '60'). To the right of each input field is a green arrow labeled with a number: 1, 2, and 3 respectively. Below these fields is a blue 'Add' button, which has a green arrow labeled 4 pointing to it. Underneath the 'Add' button is a grid table with columns: ID, Name, and Credit Hours. The table contains five rows of data. Each row has an 'Edit' and 'Delete' link in the first column. To the left of the grid, two green arrows labeled 6 and 7 point upwards towards the bottom-left corner of the input fields.

ID	Name	Credit Hours
Edit Delete CC12	Data Structure and Algorithm	60
Edit Delete CC49	Engineering Thermodynamic	60
Edit Delete SG101	Software Engineer	30
Edit Delete TG405	Data Analysis	50
Edit Delete RD609	Research and Development	60

Figure 55: Module Form

1. Enter the code of module to be added in this text field. This text field cannot be left empty.
2. Enter the name of module to be added in this text field. This text field cannot be left empty.
3. Enter the Credit hours of the module in numeric form. This text field cannot be left empty.
4. Press the Add button to add data.
5. The module details data is displayed in this grid.
6. Press this button to edit data in the grid.
7. Press this button to delete data in the grid.

12.6. Department Form

The screenshot shows a web-based application interface for managing departments. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, and others. Below the navigation bar, a text input field labeled "Department" is shown with a red border and a small circular icon in the top right corner. A green arrow labeled "1" points to this field. To the right of the input field is a blue "Add" button, which has a green arrow labeled "2" pointing to it. Below these elements is a data grid with a header row containing "ID" and "Department". The data grid contains five rows of department details:

ID	Department
1	RTE
2	Finance
3	Student Services
4	External Partnership
5	Student Career

A green arrow labeled "3" points to the left side of the data grid. At the bottom of the grid, there are two sets of "Edit" and "Delete" buttons, each with a green arrow: one set is labeled "4" and the other is labeled "5".

Figure 56: Department Form

1. Enter the name of the department to be added in this text field. This text field cannot be left empty.
2. Press the Add button to add data.
3. The department details data is displayed in this grid.
4. Press this button to edit data in the grid.
5. Press this button to delete data in the grid.

12.7. Student Form

The screenshot shows a web-based student form. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there are four text input fields labeled 'Name' (1), 'Street Number' (2), 'Street Name' (3), and 'State Name' (4). To the right of each input field is a green arrow pointing to its corresponding number. Below these fields is a blue 'Add' button, which also has a green arrow pointing to it and is labeled '5'. Underneath the 'Add' button is a grid table containing student data. The grid has columns for ID, Name, Street Number, Street Name, and State Name. Each row in the grid contains an 'Edit' and 'Delete' link in the first column. To the left of the first two rows in the grid, there are two green arrows pointing up to them, labeled '7' and '8' respectively. To the right of the grid, a large green arrow points to it and is labeled '6'.

	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Philip Morow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Dalibor Kiculski	3771	Chapler House	North Carolina

Figure 57: Student Form

1. Enter the name of the student to be added in this text field. This text field cannot be left empty.
2. Enter the street number of the student's address to be added in this text field in numeric form. This text field cannot be left empty.
3. Enter the street name of the student's address to be added in this text field. This text field cannot be left empty.
4. Enter the state name of the student's address to be added in this text field. This text field cannot be left empty.
5. Press the Add button to add data.
6. The student details data is displayed in this grid.
7. Press this button to edit data in the grid.
8. Press this button to delete data in the grid.

12.8. Teacher-Module Form

Table of Teachers

	ID	Name	EMAIL	Student ID
Select	1	Ethan Cunningham	Ethan_Cunningham529@kenn.org	
Select	2	Tyson McCall	Tyson_Mccall3864@deon.tech	
Select	3	Alexander Lee	Alexander_Lee2643@veran.org	
Select	4	Gil Garner	Gil_Garner1739@wejet.com	
Select	5	Carmen Andrews	Carmen_Andrews5864@typi.biz	
Select	6	Peter Graham	Peter_Graham5480@bulafly.com	
Select	7	Anthony Gates	Anthony_Gates1413@guentalis	
Select	8	Carl Alexander	Carl_Alexander3633@cisperito.com	
Select	9	Alesia Underhill	Alesia_Underhill02045@gmail.com	
Select	10	Kimberly Payne	Kimberly_Payne1227@latin.tech	

Table of Associated Modules

Module Code	Teacher ID	Module Name	Credit Hours
CC12	3	Data Structure and Algorithm	60
RD609	3	Research and Development	60

Figure 58: Teacher-Module Form

1. The teacher details data is displayed in this grid.
2. Press this button to select this teacher data and display the modules taught by this teacher.
3. The module details data is displayed in this grid.

12.9. Student-Fees Form

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Roadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7089	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Fee Details

Student ID	Semester	Department Name	Fee Amount	Fee Status
2	1	Finance	650	Paid
2	2	Finance	550	Paid
2	3	Finance	600	Due

Figure 59: Student_Fees Form

1. The student details data is displayed in this grid.
2. Press this button to select this student data and display the fee details of this student.
3. The fee details data is displayed in this grid.

12.10. Student-Assignment Form

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Roadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7089	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Philip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Assignment Details

Student ID	Module Code	Module Name	Assignment Type	Department Name	Grade	Status
3	RD609	Research and Development	Written Exam	RTE	A+	Pass
3	RD609	Research and Development	Unseen Exam	RTE	B	Pass
3	CC49	Engineering Thermodynamic	Individual Assignment	RTE	C+	Pass
3	CC49	Engineering Thermodynamic	Group Assignment	RTE	D	Fail

Figure 60: Student_Assignment Form

1. The student details data is displayed in this grid.
2. Press this button to select this student data and display the assignment details of this student.
3. The assignment details data is displayed in this grid.

13. Testing

13.1. Basic Forms

13.1.1. Teacher Form

13.1.1.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the teacher form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 16: Teacher Test 1

	ID	Name	EMAIL	Student ID
Edit Delete	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Edit Delete	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit Delete	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit Delete	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit Delete	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit Delete	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit Delete	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit Delete	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit Delete	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit Delete	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	

Figure 61: Teacher Test 1 evidence

13.1.1.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the teacher form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 17: Teacher Test 2

The screenshot shows a web-based application interface for managing teacher data. At the top, there is a navigation bar with a large red 'B' logo and links for Teacher, Address, Module, Department, Student, Teacher-Module, and Student-Fees. Below the navigation bar, there are input fields for Name ('Hari Man Sing') and Email ('h.man@gmail.com'), both of which have green checkmarks indicating they are valid. A dropdown menu for Student_ID is set to 'Null'. A blue 'Add' button is visible. Below these inputs is a table listing ten teacher records:

	ID	Name	Email	Student ID
Edit	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Edit	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	

Figure 62: Teacher Test 2 Added Data

The screenshot shows a web-based application interface. At the top, there is a navigation bar with a large red 'B' logo on the left, followed by menu items: Teacher, Address, Module, Department, Student, | Teacher-Module, Student-Fees, and Student-ID. Below the navigation bar, there is a form for adding a teacher. The form fields include 'Name' (with a red error icon), 'Email' (with a red error icon), and 'Student_ID' (set to 'Null'). A blue 'Add' button is located below the form. Below the form is a table listing student information:

	ID	Name	EMAIL	Student ID
Edit	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Edit	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	
Edit	11	Hari Man Sing	h.man@gmail.com	

Figure 63: Teacher Test 2 Addition Result

13.1.1.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	<ul style="list-style-type: none"> The add button disappears and an update button appears. The data in the row gets filled in the available textboxes. 	<ul style="list-style-type: none"> The add button disappeared and an update button appeared. The data in the field got filled in the available textboxes.
Click the update button	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data. 	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data.

Table 18: Teacher Test 3

The screenshot shows a web-based application interface. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there are three input fields: 'Name' containing 'Hari Man Sing' with a green checkmark, 'Email' containing 'h.man@gmail.com' with a green checkmark, and 'Student_ID' with a dropdown menu showing 'Null'. Below these fields is a table with 11 rows, each representing a student. The columns are labeled 'ID', 'Name', 'EMAIL', and 'Student ID'. Each row has 'Edit' and 'Delete' links in the first column. The data in the table is as follows:

	ID	Name	EMAIL	Student ID
Edit Delete	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Edit Delete	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit Delete	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit Delete	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit Delete	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit Delete	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit Delete	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit Delete	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit Delete	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit Delete	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	
Edit Delete	11	Hari Man Sing	h.man@gmail.com	

At the bottom left of the form area, there is a blue 'Update' button.

Figure 64: Teacher Test 3 condition 1

	ID	Name	EMAIL	Student ID
Edit	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Edit	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	
Edit	11	Hari Man Sing	h.man@gmail.com	

[Update](#)

Figure 65: Edited value in Email Textbox

The screenshot shows a web-based application interface. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, | Teacher-Module, Student-Fees, Student-Assignment. On the left, there is a large red letter 'B'. Below the navigation bar, there are input fields for 'Name' and 'Email', both with red borders and exclamation marks indicating required fields. A dropdown menu for 'Student_ID' is set to 'Null'. A blue 'Add' button is located below these fields. Below the button is a table listing 11 student records:

	ID	Name	EMAIL	Student ID
Edit	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Delete	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Delete	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Delete	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Delete	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Delete	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	
Edit	11	Hari Man Sing	hari.man@gmail.com	

Figure 66: Teacher Test 3 Condition 2

13.1.1.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 19: Teacher Test 4

The screenshot shows a web-based application interface. At the top, there is a navigation bar with a large red 'B' logo, followed by links: Teacher, Address, Module, Department, Student | Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there is a form for adding a teacher. The form fields include 'Name' (input field with a red border and an exclamation mark icon), 'Email' (input field with a red border and an exclamation mark icon), and 'Student_ID' (dropdown menu set to 'Null'). A blue 'Add' button is located below the form. To the right of the form is a data grid table with columns: ID, Name, EMAIL, and Student ID. The table contains 11 rows of teacher data, each with 'Edit' and 'Delete' links in the first column. The data in the grid is as follows:

	ID	Name	EMAIL	Student ID
Edit Delete	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Edit Delete	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit Delete	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit Delete	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit Delete	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit Delete	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit Delete	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit Delete	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit Delete	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit Delete	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	
Edit Delete	11	Hari Man Sing	hari.man@gmail.com	

Figure 67: Teacher Grid Pre-Deletion

The screenshot shows a web-based application interface for managing teacher data. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there are input fields for Name and Email, both with red validation circles containing an exclamation mark. A dropdown menu for Student_ID is set to Null. A blue 'Add' button is visible. The main area contains a table with ten rows of teacher information. The last row (ID 10) is visually distinct, suggesting it has been deleted from the database.

	ID	Name	EMAIL	Student ID
Edit Delete	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Edit Delete	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Edit Delete	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Edit Delete	4	Gil Garner	Gil_Garner8738@twipet.com	
Edit Delete	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Edit Delete	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Edit Delete	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Edit Delete	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Edit Delete	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Edit Delete	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	

Figure 68: Teacher Grid Post-Deletion

13.1.2. Address Form

13.1.2.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the address form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 20: Address Test 1

The screenshot shows a web page titled "localhost:44370/Address.aspx". The page has a header with back, forward, and refresh buttons. Below the header is a search bar with placeholder text "Street Number" and a dropdown arrow icon. Underneath the search bar are three input fields labeled "Street Name", "State Name", and a large "Add" button. Below these controls is a table listing 12 address entries. Each entry includes an "Edit Delete" link, an ID number, a street number, a street name, and a state name.

	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas

Figure 69: Address Test 1 evidence

13.1.2.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the address form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 21: Address Test 2

The screenshot shows a web application interface for managing addresses. At the top, there is a navigation bar with a large red 'B' logo and links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, there is a form with three input fields: 'Street Number' containing '96' with a green checkmark, 'Street Name' containing 'Black Lack', and 'State Name' containing 'Montana'. Below the form is a blue 'Add' button. To the right of the form is a data grid table with columns: ID, Street Number, Street Name, and State Name. The table contains two rows of data:

	ID	Street Number	Street Name	State Name
Edit	1	4000	Littlebury Street	Texas
Edit	2	4251	Sheffield Walk	Hawaii

Figure 70: Address Test 2 Data to be Added

The screenshot shows a web browser window with the URL `localhost:44370/Address.aspx`. The page displays a table of address data and a form for adding new addresses.

Form Fields:

- Street Name:** A text input field with a red border and a red exclamation mark icon.
- State Name:** A text input field with a red border and a red exclamation mark icon.
- Add:** A blue button.

Table Data:

	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	Black Lack	Montana

Figure 71: Address Test 2 Addition Result

13.1.2.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	<ul style="list-style-type: none"> The add button disappears and an update button appears. The data in the row gets filled in the available textboxes. 	<ul style="list-style-type: none"> The add button disappeared and an update button appeared. The data in the field got filled in the available textboxes. However, the street number got loaded into all textboxes

Table 22: Address Test 3 Fail Case

The screenshot shows a web page titled "localhost:44370/Address.aspx". At the top, there are three input fields: "Street Number" (containing "96" with a green checkmark), "Street Name" (containing "96" with a green checkmark), and "State Name" (containing "96" with a green checkmark). Below these is a table with columns: ID, Street Number, Street Name, and State Name. The table contains 14 rows of data. The last row (ID 14) has a different background color. At the bottom left is a blue "Update" button, and at the bottom center is a piece of JavaScript code.

	ID	Street Number	Street Name	State Name
Edit	1	4000	Littlebury Street	Texas
Edit	2	4251	Sheffield Walk	Hawaii
Edit	3	8434	Thoresby Rue	North Carolina
Edit	4	6808	Argyle Crossroad	Virginia
Edit	5	5532	Thrale Drive	Massachusetts
Edit	6	452	Epworth Way	Colorado
Edit	7	1268	Chart Route	New Hampshire
Edit	8	7176	South Road	Ohio
Edit	9	470	West Hill	South Carolina
Edit	10	2972	Blenkarne Avenue	North Dakota
Edit	11	8296	Queen Road	Montana
Edit	12	9902	Boadicea Drive	Arkansas
Edit	14	96	Black Lack	Montana

Update

```
javascript:_doPostBack('ctl00$ContentPlaceHolder1$addressGv','Edit$12')
```

Figure 72: Address Test 3 condition 1 fail case

13.1.2.3.1. What caused the issue and how it was rectified:

```
protected void OnRowEditing(object sender, GridViewEditEventArgs e)
{
    IDTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[1].Text;
    streetNoTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    streetNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    stateNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    updateBtn.Visible = true;
    submitBtn.Visible = false;
}
```

Figure 73: Issue causing the fail case

The textboxes were updated with the grid row cell containing the street number value. Once they were changed, relevant values got added to the respective textboxes.

```
protected void OnRowEditing(object sender, GridViewEditEventArgs e)
{
    IDTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[1].Text;
    streetNoTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[2].Text.ToString();
    streetNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[3].Text.ToString();
    stateNameTxt.Text = this.addressGv.Rows[e.NewEditIndex].Cells[4].Text.ToString();
    updateBtn.Visible = true;
    submitBtn.Visible = false;
}
```

Figure 74: Rectification of the Issue causing the fail case

The screenshot shows a web application interface for address rectification. At the top, there are three input fields: 'Street Number' (96), 'Street Name' (Black Lack), and 'State Name' (Montana), each with a green checkmark indicating success. Below these is a table listing 14 address records. Each record includes an 'Edit' and 'Delete' link, an ID, Street Number, Street Name, and State Name. The last record in the table is the one that was just rectified: ID 14, Street Number 96, Street Name Black Lack, and State Name Montana.

	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	Black Lack	Montana

Update

Figure 75: Test 3 Condition 1 post rectification

Test Condition	Expected Outcome	Obtained Outcome
Click the update button	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data. 	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data.

Table 23: Address Test 3 condition 2

96	✓
Street Name	
White Lack	✓
State Name	
Montana	✓

	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	Black Lack	Montana

Update

Figure 76: Edited value in Street Name Textbox

The screenshot shows a web browser window with the URL `localhost:44370/Address.aspx`. At the top, there are three input fields with red borders and a red exclamation mark icon in each: 'Street Name', 'State Name', and another unlabeled field. Below these is a blue 'Add' button. A table follows, containing 14 rows of address data. Each row has a 'Edit Delete' link in the first column. The columns are labeled: ID, Street Number, Street Name, and State Name.

	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	White Lack	Montana

Figure 77: Address Test 3 Condition 2

13.1.2.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 24: Address Test 4

The screenshot shows a web browser window with the URL `localhost:44370/Address.aspx`. At the top, there is a header bar with icons for back, forward, and refresh, followed by the address bar. Below the header is a form with three input fields: 'Street Name' (containing 'Littlebury Street'), 'State Name' (containing 'Texas'), and a large empty text area. A blue 'Add' button is located below the form. To the right of the form is a data grid table with 14 rows of address data. The columns are labeled 'ID', 'Street Number', 'Street Name', and 'State Name'. Each row contains an 'Edit' link and a 'Delete' link in the first column. The data in the grid is as follows:

	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas
Edit Delete	14	96	White Lack	Montana

Figure 78: Address Grid Pre-Deletion

Street Number

Street Name

State Name

Add

	ID	Street Number	Street Name	State Name
Edit Delete	1	4000	Littlebury Street	Texas
Edit Delete	2	4251	Sheffield Walk	Hawaii
Edit Delete	3	8434	Thoresby Rue	North Carolina
Edit Delete	4	6808	Argyle Crossroad	Virginia
Edit Delete	5	5532	Thrale Drive	Massachusetts
Edit Delete	6	452	Epworth Way	Colorado
Edit Delete	7	1268	Chart Route	New Hampshire
Edit Delete	8	7176	South Road	Ohio
Edit Delete	9	470	West Hill	South Carolina
Edit Delete	10	2972	Blenkarne Avenue	North Dakota
Edit Delete	11	8296	Queen Road	Montana
Edit Delete	12	9902	Boadicea Drive	Arkansas

Figure 79: Address Grid Post-Deletion

13.1.3. Module Form

13.1.3.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the module form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 25: Module Test 1

The screenshot shows a web application interface for managing modules. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, and Teacher-Module/Student. Below the navigation bar, there is a form with three input fields labeled 'Module Code', 'Name', and 'Credit Hours', each with a red-bordered input field containing a placeholder 'Enter value' and a red question mark icon. Below the form is a blue 'Add' button. To the right of the form is a table displaying a list of modules:

	ID	Name	Credit Hours
Edit Delete	CC12	Data Structure and Algorithm	60
Edit Delete	CC49	Engineering Thermodynamic	60
Edit Delete	SG101	Software Engineer	30
Edit Delete	TG405	Data Analysis	50
Edit Delete	RD609	Research and Development	60

Figure 80: Module Test 1 evidence

13.1.3.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the module form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 26: Module Test 2

The screenshot shows a web-based application interface for managing modules. At the top, there is a navigation bar with a large red 'B' logo and links for Teacher, Address, Module, Department, Student, and Teacher-Module. Below the navigation bar, the main content area has a dark header with the text "Module Code". A text input field contains "RR34" with a green checkmark icon to its right. The next section is labeled "Name" with a text input field containing "Computer Aided Drawing". Below that is a section labeled "Credit Hours" with a numeric input field containing "60", accompanied by up and down arrow buttons and a green checkmark icon. A blue "Add" button is located below these input fields. At the bottom of the page is a table listing five existing modules:

	ID	Name	Credit Hours
Edit Delete	CC12	Data Structure and Algorithm	60
Edit Delete	CC49	Engineering Thermodynamic	60
Edit Delete	SG101	Software Engineer	30
Edit Delete	TG405	Data Analysis	50
Edit Delete	RD609	Research and Development	60

Figure 81: Module Test 2 Data to be Added

The screenshot shows a web-based application interface for managing module records. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Module, and Log Out. Below the navigation bar, there are three input fields labeled 'Module Code', 'Name', and 'Credit Hours', each with a red-bordered error message box containing an exclamation mark. A blue 'Add' button is positioned below these fields. Below the button is a table listing six existing module records:

	ID	Name	Credit Hours
Edit Delete	CC12	Data Structure and Algorithm	60
Edit Delete	CC49	Engineering Thermodynamic	60
Edit Delete	SG101	Software Engineer	30
Edit Delete	TG405	Data Analysis	50
Edit Delete	RD609	Research and Development	60
Edit Delete	RR34	Computer Aided Drawing	60

Figure 82: Module Test 2 Addition Result

13.1.3.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	<ul style="list-style-type: none"> The add button disappears and an update button appears. The data in the row gets filled in the available textboxes. 	<ul style="list-style-type: none"> The add button disappeared and an update button appeared. The data in the field got filled in the available textboxes.
Click the update button	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data. 	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data.

Table 27: Module Test 3

The screenshot shows a web-based application interface for managing modules. At the top, there is a navigation bar with a large red 'B' logo and links for Teacher, Address, Module, Department, Student, and Teacher-Module. Below the navigation bar, there are two input fields: 'Name' containing 'Computer Aided Drawing' with a green checkmark, and 'Credit Hours' containing '60' with a green checkmark. Below these fields is a table listing six modules:

	ID	Name	Credit Hours
Edit Delete	CC12	Data Structure and Algorithm	60
Edit Delete	CC49	Engineering Thermodynamic	60
Edit Delete	SG101	Software Engineer	30
Edit Delete	TG405	Data Analysis	50
Edit Delete	RD609	Research and Development	60
Edit Delete	RR34	Computer Aided Drawing	60

At the bottom left of the form area is a blue 'Update' button.

Figure 83: Module Test 3 condition 1

B Teacher Address Module Department Student | Teacher-Module S

Name
Computer Aided Drawing ✓

Credit Hours
30 ✓

	ID	Name	Credit Hours
Edit	CC12	Data Structure and Algorithm	60
Edit	CC49	Engineering Thermodynamic	60
Edit	SG101	Software Engineer	30
Edit	TG405	Data Analysis	50
Edit	RD609	Research and Development	60
Edit	RR34	Computer Aided Drawing	60

Update

Figure 84: Edited value in Credit Hours Textbox

The screenshot shows a web-based application interface for managing modules. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, Teacher-Module, and Search. A large red letter 'B' is prominently displayed on the left side of the header.

The main form area contains three input fields:

- Module Code: An input field with a placeholder '(1)'.
- Name: An input field with a placeholder '(1)'.
- Credit Hours: An input field with a placeholder '(1)'.

Below the form is a blue 'Add' button.

At the bottom, there is a table listing existing module records:

	ID	Name	Credit Hours
Edit	CC12	Data Structure and Algorithm	60
Edit	CC49	Engineering Thermodynamic	60
Edit	SG101	Software Engineer	30
Edit	TG405	Data Analysis	50
Edit	RD609	Research and Development	60
Edit	RR34	Computer Aided Drawing	30

Figure 85: Module Test 3 Condition 2

13.1.3.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 28: Module Test 4

The screenshot shows a web-based application interface. At the top, there is a navigation bar with a large red 'B' logo, followed by links for Teacher, Address, Module, Department, Student, and Teacher-Module. Below the navigation bar, there is a search bar labeled 'Search' with a magnifying glass icon. The main content area has three input fields: 'Module Code' (with a red error icon), 'Name' (with a red error icon), and 'Credit Hours' (with a red error icon). A blue 'Add' button is located below these fields. To the right, there is a data grid table with columns for ID, Name, and Credit Hours. Each row in the grid contains an 'Edit' link and a 'Delete' link. The data in the grid is as follows:

	ID	Name	Credit Hours	
Edit	Delete	CC12	Data Structure and Algorithm	60
Edit	Delete	CC49	Engineering Thermodynamic	60
Edit	Delete	SG101	Software Engineer	30
Edit	Delete	TG405	Data Analysis	50
Edit	Delete	RD609	Research and Development	60
Edit	Delete	RR34	Computer Aided Drawing	30

Figure 86: Address Grid Pre-Deletion

13.1.3.4.1. What caused the issue and how it was rectified:

```

    protected void OnRowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        int ID = Convert.ToInt32(moduleGv.DataKeys[e.RowIndex].Values[0]);
        string constr = ConfigurationManager.ConnectionStrings["coursework"].ConnectionString;
        using (OracleConnection con = new OracleConnection(constr))
        {
            using (OracleCommand cmd = new OracleCommand("DELETE FROM Module"))
            {
                cmd.Connection = con;
                con.Open();
                cmd.ExecuteNonQuery();
                con.Close();
            }
        }
        this.BindGrid();
        moduleGv.EditIndex = -1;
    }

```

Figure 87: Issue causing the fail case

Unlike the primary keys from other form table, module table had a VARCHAR primary key. Here, I was trying to convert the string into an integer in order to obtain the primary key for the row of data that needed to be deleted. Hence, it caused a format exception and did not carry out the delete procedure.

I set the variable to string and used the value obtained from the grid view row as a string and rectified the issue. This made the deletion procedure go smoothly and uninterrupted.

```

    protected void OnRowDeleting(object sender, GridViewDeleteEventArgs e)
    {

        string ID = moduleGv.DataKeys[e.RowIndex].Values[0].ToString();
        string constr = ConfigurationManager.ConnectionStrings["coursework"].ConnectionString;
        using (OracleConnection con = new OracleConnection(constr))
        {
            using (OracleCommand cmd = new OracleCommand($"DELETE FROM Module WHERE Module_Code = '{ID}'"))
            {
                cmd.Connection = con;
                con.Open();
                cmd.ExecuteNonQuery();
                con.Close();
            }
        }
    }

```

Figure 88: Rectification of the Issue causing the fail case

Module Code

Name

Credit Hours

Add

	ID	Name	Credit Hours
Edit Delete	CC12	Data Structure and Algorithm	60
Edit Delete	CC49	Engineering Thermodynamic	60
Edit Delete	SG101	Software Engineer	30
Edit Delete	TG405	Data Analysis	50
Edit Delete	RD609	Research and Development	60

Figure 89: Test 4 post deletion

13.1.4. Department Form

13.1.4.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the department form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 29: Department Test 1

	ID	Department
Edit	1	RTE
Edit	2	Finance
Edit	3	Student Services
Edit	4	External Partnership
Edit	5	Student Career

Figure 90: Department Test 1 evidence

13.1.4.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the department form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 30: Department Test 2

The screenshot shows a web-based application interface. At the top, there is a navigation bar with a large red 'B' icon followed by links: Teacher, Address, Module, Department, Student, and Home. Below the navigation bar, the word "Department" is displayed in blue text. A text input field contains the value "Counseling" with a green checkmark icon to its right, indicating it has been added. A blue "Add" button is located below the input field. To the right, there is a table with the following data:

	ID	Department
Edit	1	RTE
Edit	2	Finance
Edit	3	Student Services
Edit	4	External Partnership
Edit	5	Student Career

Figure 91: Department Test 2 Added Data

The screenshot shows a web-based application interface. At the top, there is a dark header bar with a large red letter 'B' on the left and several menu items: Teacher, Address, Module, Department, and Student. Below the header, the word "Department" is displayed in blue text. Underneath this, there is a red-bordered input field containing a red exclamation mark (!). A blue button labeled "Add" is positioned below the input field. The main content area contains a table with six rows, each representing a department. The columns are labeled "Edit Delete", "ID", and "Department". The data in the table is as follows:

	ID	Department
Edit Delete	1	RTE
Edit Delete	2	Finance
Edit Delete	3	Student Services
Edit Delete	4	External Partnership
Edit Delete	5	Student Career
Edit Delete	6	Counseling

Figure 92: Department Test 2 Addition Result

13.1.4.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	<ul style="list-style-type: none"> The add button disappears and an update button appears. The data in the row gets filled in the available textboxes. 	<ul style="list-style-type: none"> The add button disappeared and an update button appeared. The data in the field got filled in the available textboxes.
Click the update button	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data. 	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data.

Table 31: Department Test 3

The screenshot shows a web-based application interface. At the top, there is a navigation bar with a large red 'B' logo and links for Teacher, Address, Module, Department, Student, and Log Out. Below the navigation bar, a search bar contains the text 'Department' and a dropdown menu with the option 'Counseling' selected, indicated by a green checkmark. The main content area displays a table of department data:

	ID	Department
Edit Delete	1	RTE
Edit Delete	2	Finance
Edit Delete	3	Student Services
Edit Delete	4	External Partnership
Edit Delete	5	Student Career
Edit Delete	6	Counseling

At the bottom left of the main content area is a blue 'Update' button.

Figure 93: Department Test 3 condition 1

The screenshot shows a web-based application interface. At the top left is a large red letter 'B'. To its right are navigation links: Teacher, Address, Module, Department, and others partially visible. Below this is a search bar labeled 'Department' containing the text 'Cognitive Counseling' with a green checkmark icon to its right. The main content area displays a table with six rows, each having 'Edit' and 'Delete' buttons in the first column. The columns are labeled 'ID' and 'Department'. The data is as follows:

	ID	Department
Edit Delete	1	RTE
Edit Delete	2	Finance
Edit Delete	3	Student Services
Edit Delete	4	External Partnership
Edit Delete	5	Student Career
Edit Delete	6	Counseling

At the bottom left of the form is a blue 'Update' button.

Figure 94: Edited value in Department Textbox

	ID	Department
Edit Delete	1	RTE
Edit Delete	2	Finance
Edit Delete	3	Student Services
Edit Delete	4	External Partnership
Edit Delete	5	Student Career
Edit Delete	6	Cognitive Counseling

Figure 95: Department Test 3 Condition 2

13.1.4.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 32: Department Test 4

	ID	Department
Edit	1	RTE
Edit	2	Finance
Edit	3	Student Services
Edit	4	External Partnership
Edit	5	Student Career
Edit	6	Cognitive Counseling

Figure 96: Department Grid Pre-Deletion

	ID	Department	
Edit Delete	1	RTE	
Edit Delete	2	Finance	
Edit Delete	3	Student Services	
Edit Delete	4	External Partnership	
Edit Delete	5	Student Career	

Figure 97: Department Grid Post-Deletion

13.1.5. Student Form

13.1.5.1. Test 1: View

Test Condition	Expected Outcome	Obtained Outcome
Load the student form and check if data is loaded in the grid.	The data is loaded on page load.	The data was loaded on page load.

Table 33: Student Test 1

ID	Name	Street Number	Street Name	State Name
1	Michelle Grady	6342	Baylis Rue	Massachusetts
2	Sara Anderson	9896	Baylis Drive	New Mexico
3	Cherish Leigh	8296	Queen Road	Montana
4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
5	Sydney London	5238	South Avenue	North Dakota
6	Hannah Giles	7889	King Edward Route	Michigan
7	Jack Hammond	9827	Ensign Crossroad	Idaho
8	Phillip Morrow	7410	Garfield Grove	Alabama
9	Carissa Clarke	8580	Buttonwood Way	Virginia
10	Deborah Bingham	3771	Chester Grove	North Carolina

Figure 98: Student Test 1 evidence

13.1.5.2. Test 2: Add

Test Condition	Expected Outcome	Obtained Outcome
Fill the student form appropriately and press the add button.	The data is added, and the grid is refreshed with the new data.	The data was added without any errors and the grid was refreshed with the new data.

Table 34: Student Test 2

The screenshot shows a web-based application interface. At the top, there is a navigation bar with a large red 'B' logo and links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Grid. Below the navigation bar, there are four input fields with dropdown menus: 'Name' (Bir Bahadur Karki), 'Street Number' (9669), 'Street Name' (King Road), and 'State Name' (Idaho). Each of these input fields has a green checkmark icon to its right. Below these fields is a blue 'Add' button. To the right of the input fields is a data grid table with the following columns: ID, Name, Street Number, Street Name, and State Name. The table contains 10 rows of data, each with 'Edit' and 'Delete' links in the first column. The data in the grid is as follows:

	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Deborah Bingham	3771	Chester Grove	North Carolina

Figure 99: Student Test 2 Added Data

Name	<input type="text"/>				
Street Number	<input type="text"/>				
Street Name	<input type="text"/>				
State Name	<input type="text"/>				
<input type="button" value="Add"/>					
	ID	Name	Street Number	Street Name	State Name
Edit	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit	3	Cherish Leigh	8296	Queen Road	Montana
Edit	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit	5	Sydney London	5238	South Avenue	North Dakota
Edit	6	Hannah Giles	7889	King Edward Route	Michigan
Edit	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit	10	Deborah Bingham	3771	Chester Grove	North Carolina
Edit	11	Bir Bahadur Karki	9669	King Road	Idaho

Figure 100: Student Test 2 Addition Result

13.1.5.3. Test 3: Edit

Test Condition	Expected Outcome	Obtained Outcome
Click the edit button	<ul style="list-style-type: none"> The add button disappears and an update button appears. The data in the row gets filled in the available textboxes. 	<ul style="list-style-type: none"> The add button disappeared and an update button appeared. The data in the field got filled in the available textboxes.
Click the update button	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data. 	<ul style="list-style-type: none"> The update button disappears and the add button reappears The data in the grid is refreshed with the edited data.

Table 35: Student Test 3

Name
 ✓

Street Number
 ✓

Street Name
 ✓

State Name
 ✓

	ID	Name	Street Number	Street Name	State Name
Edit	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit	3	Cherish Leigh	8296	Queen Road	Montana
Edit	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit	5	Sydney London	5238	South Avenue	North Dakota
Edit	6	Hannah Giles	7889	King Edward Route	Michigan
Edit	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit	10	Deborah Bingham	3771	Chester Grove	North Carolina
Edit	11	Bir Bahadur Karki	9669	King Road	Idaho

Figure 101: Student Test 3 condition 1

Name					
Bir Bahadur Karki	✓				
Street Number					
6996	✓				
Street Name					
King Road	✓				
State Name					
Idaho	✓				
	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Deborah Bingham	3771	Chester Grove	North Carolina
Edit Delete	11	Bir Bahadur Karki	9669	King Road	Idaho

Update

Figure 102: Edited value in Street Number Textbox

Name	<input type="text"/>				
Street Number	<input type="text"/>				
Street Name	<input type="text"/>				
State Name	<input type="text"/>				
Add					
	ID	Name	Street Number	Street Name	State Name
Edit	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit	3	Cherish Leigh	8296	Queen Road	Montana
Edit	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit	5	Sydney London	5238	South Avenue	North Dakota
Edit	6	Hannah Giles	7889	King Edward Route	Michigan
Edit	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit	10	Deborah Bingham	3771	Chester Grove	North Carolina
Edit	11	Bir Bahadur Karki	6996	King Road	Idaho

Figure 103: Student Test 3 Condition 2

13.1.5.4. Test 4: Delete

Test Condition	Expected Outcome	Obtained Outcome
Click the delete button.	The grid is refreshed with the data row deleted.	The grid was refreshed with the data row deleted.

Table 36: Student Test 4

Name

Street Number

Street Name

State Name

Add

	ID	Name	Street Number	Street Name	State Name	
Edit	1	Michelle Grady	6342	Baylis Rue	Massachusetts	Delete
Edit	2	Sara Anderson	9896	Baylis Drive	New Mexico	Delete
Edit	3	Cherish Leigh	8296	Queen Road	Montana	Delete
Edit	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas	Delete
Edit	5	Sydney London	5238	South Avenue	North Dakota	Delete
Edit	6	Hannah Giles	7889	King Edward Route	Michigan	Delete
Edit	7	Jack Hammond	9827	Ensign Crossroad	Idaho	Delete
Edit	8	Phillip Morrow	7410	Garfield Grove	Alabama	Delete
Edit	9	Carissa Clarke	8580	Buttonwood Way	Virginia	Delete
Edit	10	Deborah Bingham	3771	Chester Grove	North Carolina	Delete
Edit	11	Bir Bahadur Karki	6996	King Road	Idaho	Delete

Figure 104: Student Grid Pre-Deletion

The screenshot shows a web application interface for managing student addresses. At the top, there is a navigation bar with links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assigned. The URL in the browser is `localhost:44370/Student.aspx`. Below the navigation bar, there are four input fields labeled 'Name', 'Street Number', 'Street Name', and 'State Name', each with a red border and a small info icon (i) next to it. A blue 'Add' button is located below these fields. Below the form is a data grid table with the following columns: ID, Name, Street Number, Street Name, and State Name. The table contains 10 rows of data, each with 'Edit' and 'Delete' links in the first column. The data is as follows:

	ID	Name	Street Number	Street Name	State Name
Edit Delete	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Edit Delete	2	Sara Anderson	9896	Baylis Drive	New Mexico
Edit Delete	3	Cherish Leigh	8296	Queen Road	Montana
Edit Delete	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Edit Delete	5	Sydney London	5238	South Avenue	North Dakota
Edit Delete	6	Hannah Giles	7889	King Edward Route	Michigan
Edit Delete	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Edit Delete	8	Phillip Morrow	7410	Garfield Grove	Alabama
Edit Delete	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Edit Delete	10	Deborah Bingham	3771	Chester Grove	North Carolina

Figure 105: Student Grid Post-Deletion

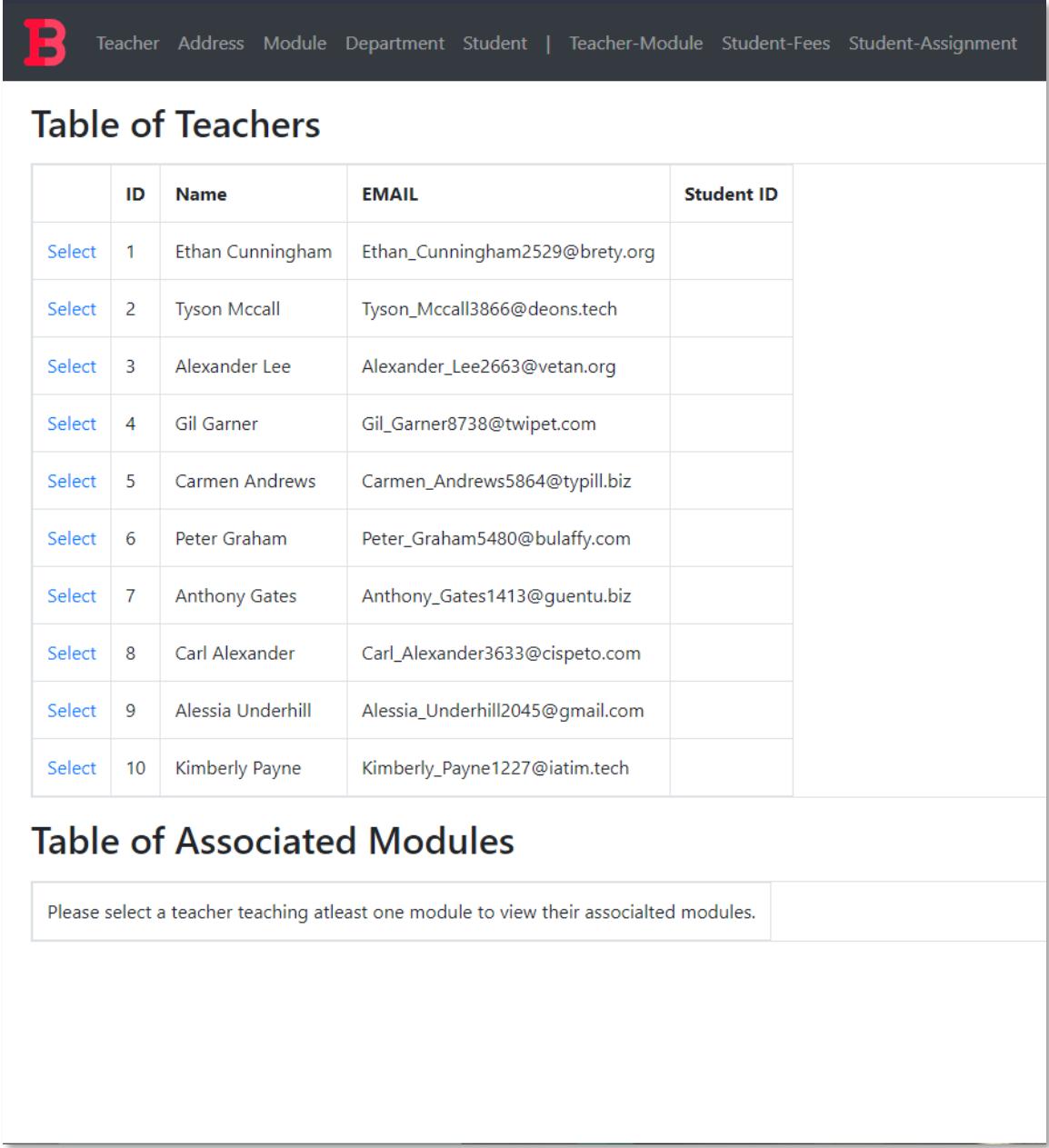
13.2. Complex Forms

13.2.1. Teacher-Module Mapping Form

13.2.1.1. Test 1: Page Load

Test Condition	Expected Outcome	Obtained Outcome
Load the complex form and check if data is loaded in the initial grid.	The data is loaded to the initial grid on page load.	The data was loaded to the initial grid on page load.

Table 37: Teacher-Module Test 1



The screenshot shows a web-based database application interface. At the top, there is a navigation bar with a large red 'B' logo on the left, followed by links: Teacher, Address, Module, Department, Student, | Teacher-Module, Student-Fees, Student-Assignment. Below the navigation bar, the title 'Table of Teachers' is displayed in bold black font. A table follows, with columns: ID, Name, EMAIL, and Student ID. The table contains 10 rows, each representing a teacher with a 'Select' link in the first column.

	ID	Name	EMAIL	Student ID
Select	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Select	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Select	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Select	4	Gil Garner	Gil_Garner8738@twipet.com	
Select	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Select	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Select	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Select	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Select	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Select	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	

Table of Associated Modules

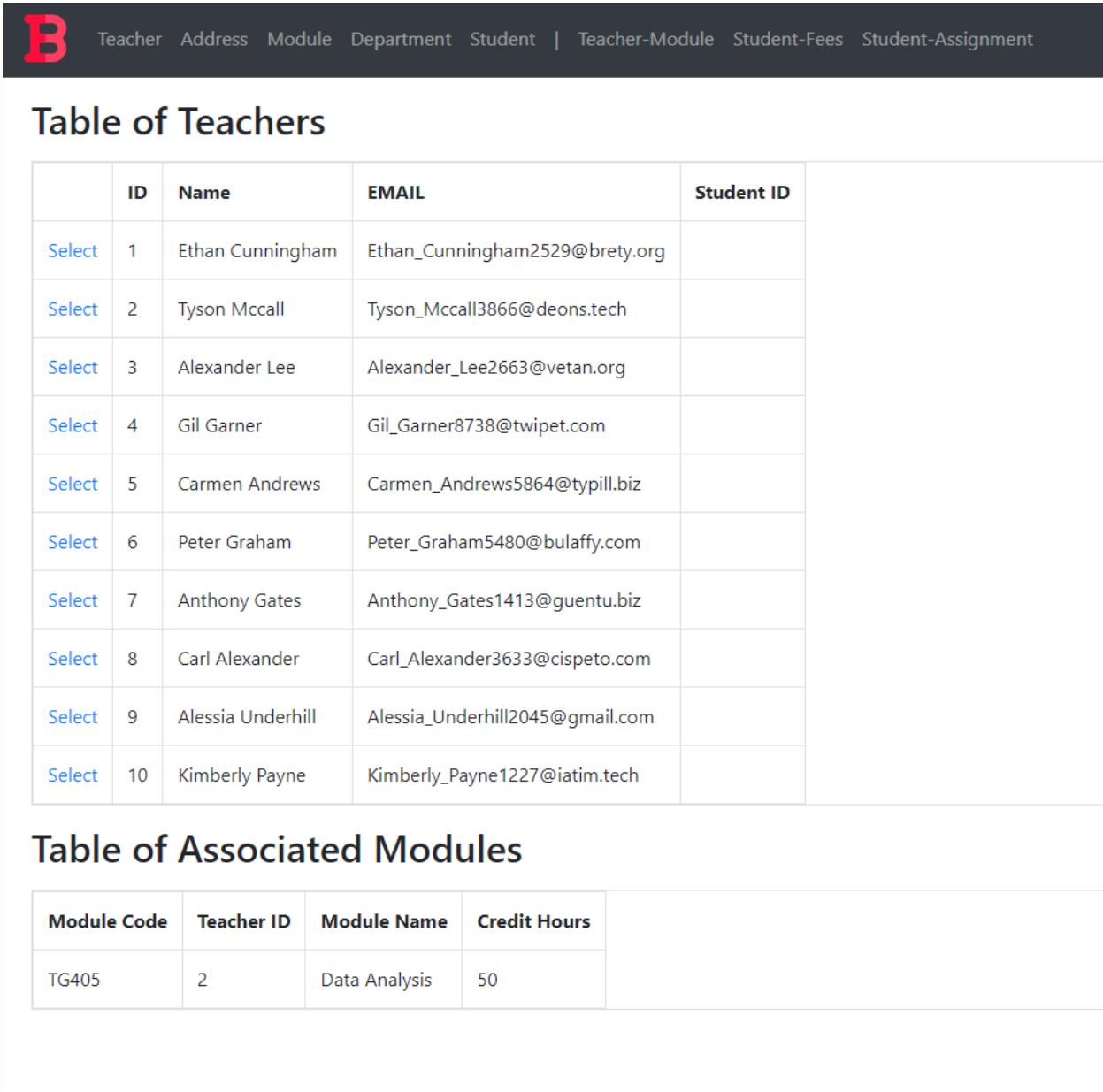
Please select a teacher teaching atleast one module to view their assocalted modules.

Figure 106: Teacher-Module Test 1 evidence

13.2.1.2. Test 2: Data Load on Selection

Test Condition	Expected Outcome	Obtained Outcome
Select a data row from the initial grid view and check if the other grid gets loaded with relevant data.	The other grid is loaded with relevant data.	The other grid was loaded with relevant data.

Table 38: Teacher-Module Test 2



The screenshot shows a web-based database interface. At the top, there is a navigation bar with links: Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, the title "Table of Teachers" is displayed. A table lists ten teachers with their ID, name, email, and student ID (which is blank). The teacher IDs range from 1 to 10, and their names and emails correspond to the "Select" row numbers. Below this, the title "Table of Associated Modules" is shown, followed by a table with four columns: Module Code, Teacher ID, Module Name, and Credit Hours. One entry is present, showing TG405, Teacher ID 2, Data Analysis, and 50 credit hours.

	ID	Name	EMAIL	Student ID
Select	1	Ethan Cunningham	Ethan_Cunningham2529@brety.org	
Select	2	Tyson Mccall	Tyson_Mccall3866@deons.tech	
Select	3	Alexander Lee	Alexander_Lee2663@vetan.org	
Select	4	Gil Garner	Gil_Garner8738@twipet.com	
Select	5	Carmen Andrews	Carmen_Andrews5864@typill.biz	
Select	6	Peter Graham	Peter_Graham5480@bulaffy.com	
Select	7	Anthony Gates	Anthony_Gates1413@guentu.biz	
Select	8	Carl Alexander	Carl_Alexander3633@cispeto.com	
Select	9	Alessia Underhill	Alessia_Underhill2045@gmail.com	
Select	10	Kimberly Payne	Kimberly_Payne1227@iatim.tech	

Module Code	Teacher ID	Module Name	Credit Hours
TG405	2	Data Analysis	50

Figure 107: Teacher-Module Test 2 evidence

13.2.2. Student-Fees Mapping Form

13.2.2.1. Test 1: Page Load

Test Condition	Expected Outcome	Obtained Outcome
Load the complex form and check if data is loaded in the initial grid.	The data is loaded to the initial grid on page load.	The data was loaded to the initial grid on page load.

Table 39: Student-Fees Test 1

The screenshot shows a web application interface. At the top, there is a dark header bar with a large red letter 'B' logo on the left, followed by navigation links: Teacher, Address, Module, Department, Student, | Teacher-Module, Student-Fees, Student-Assignment. Below the header, the main content area has a title 'Table of Student' and a table with 10 rows of data. After the table, there is another section titled 'Table of Fee Details' containing a message box.

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Fee Details

Please select a student who has been assigned at least one fee invoice.

Figure 108: Student-Fees Test 1 evidence

13.2.2.2. Test 2: Data Load on Selection

Test Condition	Expected Outcome	Obtained Outcome
Select a data row from the initial grid view and check if the other grid gets loaded with relevant data.	The other grid is loaded with relevant data.	The other grid was loaded with relevant data.

Table 40: Student-Fees Test 2

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Fee Details

Student ID	Semester	Department Name	Fee Amount	Fee Status
8	1	Finance	650	Paid
8	2	Finance	550	Paid
8	3	Finance	600	Paid
9	4	Finance	650	Paid

Figure 109: Student-Fees Test 2 evidence

13.2.3. Student-Assignment Mapping Form

13.2.3.1. Test 1: Page Load

Test Condition	Expected Outcome	Obtained Outcome
Load the complex form and check if data is loaded in the initial grid.	The data is loaded to the initial grid on page load.	The data was loaded to the initial grid on page load.

Table 41: Student-Assignment Test 1

The screenshot shows a web application interface. At the top, there is a navigation bar with a large red 'B' logo and links for Teacher, Address, Module, Department, Student, Teacher-Module, Student-Fees, and Student-Assignment. Below the navigation bar, the title 'Table of Student' is displayed. A table with 10 rows of student data is shown. The columns are labeled: ID, Name, Street No., Street Name, and State Name. The data is as follows:

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Below the table, the title 'Table of Assignment Details' is shown. A message box contains the text: 'Please select a student who has been graded on atleast one assignment.'

Figure 110: Student-Assignment Test 1 evidence

13.2.3.2. Test 2: Data Load on Selection

Test Condition	Expected Outcome	Obtained Outcome
Select a data row from the initial grid view and check if the other grid gets loaded with relevant data.	The other grid is loaded with relevant data.	The other grid was loaded with relevant data.

Table 42: Student-Assignment Test 2

Table of Student

	ID	Name	Street No.	Street Name	State Name
Select	1	Michelle Grady	6342	Baylis Rue	Massachusetts
Select	2	Sara Anderson	9896	Baylis Drive	New Mexico
Select	3	Cherish Leigh	8296	Queen Road	Montana
Select	4	Sebastian Ballard	9902	Boadicea Drive	Arkansas
Select	5	Sydney London	5238	South Avenue	North Dakota
Select	6	Hannah Giles	7889	King Edward Route	Michigan
Select	7	Jack Hammond	9827	Ensign Crossroad	Idaho
Select	8	Phillip Morrow	7410	Garfield Grove	Alabama
Select	9	Carissa Clarke	8580	Buttonwood Way	Virginia
Select	10	Deborah Bingham	3771	Chester Grove	North Carolina

Table of Assignment Details

Student ID	Module Code	Module Name	Assignment Type	Department Name	Grade	Status
3	RD609	Research and Development	Written Exam	RTE	A+	Pass
3	RD609	Research and Development	Unseen Exam	RTE	B	Pass
3	CC49	Engineering Thermodynamic	Individual Assignment	RTE	C+	Pass
3	CC49	Engineering Thermodynamic	Coursework	RTE	E+	Fail

Figure 111: Student-Assignment Test 2 evidence

14. Further Discussion

The project was a simple one, but it provided a lot of insight on oracle databases and dotnet webforms. While the database normalization, integration with assumptions, creation and data population portions were familiar to what we had already been taught, the dotnet webform creation was a unique experience in this project. As dotnet was used previously for another module to create a windows form application, I was already familiar with the dotnet syntax. Similarly, web development was also quite familiar since I had experience working with JavaScript frontend frameworks like React.

However, web development with dotnet was almost completely different than what I had experience with. The project taught me about real world business scenarios. I learned how normalization, being the useful tool that it is, sometimes does not give what we need, about how redundancy must be sometimes added artificially to keep record of things. The project also helped me better understand dotnet as a framework for web development, about how master classes can be used for page layouts, about how libraries like bootstrap can be used to visually enhance a webpage without too much effort and about HTML5 technologies such as input validation. The project also taught me about Oracle SQL sequences and triggers for auto generation of keys, about how Oracle Developer and Oracle Developer Data Modeler can be used to visualize databases, easily edit, or create tables, and generate DDL scripts. It also deepened my understanding about user permissions and Oracle database connection with third parties such as visual studio.

15. Conclusion

The coursework completion fueled my feel confidence to create a database based on real world business scenario to store data in a professional manner. Since the coursework was about creating a database record system and a web application for a college, it has lots of applicability in the real world as most colleges and schools use a database to record their data. Not only schools but a lot of management systems are built the same way with different business scenarios.

All in all, the help from teachers and timely self-research turned the project from what I believe would be intimidating to a simple and fun project that taught me a ton of techniques. While the project was a simple one it taught me more about oracle, dotnet for web development and more about problem solving in general.