**Implementation Details**

We haven’t used Triggers, Indices or Views in our implementation of the project. We have followed all the design principals and Normalized our tables. We did not felt the need to use Triggers or Indices or Views as we were able to fetch required values using simple and complex queries. We have implemented 3 complex queries as per instructions and were successfully able to display the expected data.

Following is our Relational Schema, which has main and respective bridge table.

**Students**

We used artificial key StudentEmailAddress for this table’s primary key, as no other attribute about a Student could uniquely identify a tuple.

Because of this, and because we are required to have StudentName to store customer information, StudentEmailAddress and StudentName have the constraint that it can’t be null.

**Courses**

We used artificial key CourseID for this table’s primary key, as no other attribute about a Course could uniquely identify a tuple.

Because of this, and because we are required to have CourseName and CourseOverview to store basic information about Courses, CourseID, CourseName and CourseOverview have the constraint thait it can’t be null

**Universities**

We used artificial key UniversityID for this table’s primary key, as no other attribute about a University could uniquely identify a tuple.

Because of this, and because we are required to have UniversityName to hold a University, UniversityID and UniversityName could not be null.

**Instructors**

We used artificial key InstructorID for this table’s primary key, as no other attribute about an Insturctor could uniquely identify a tuple.

Because of this, and because we are required to have InstructorName to hold an Instructor, InstructorID and InstructorName could not be null.

**Transactions**

We used artificial key TransactionID for this table’s primary key, as no other attribute about an Insturctor could uniquely identify a tuple.

Because of this, and because we are required to have TransactionTimeStamp and TransactionStatus to hold an Instructor, TransactionID, TransactionTimeStamp and TransactionStatus could not be null.

**CreditCards**

We used artificial key CreditCardNo for this table’s primary key, as no other attribute about an Insturctor could uniquely identify a tuple.

Because of this, and because we are required to have ExpDate, NameOnCard and CVV to hold an Instructor, InstructorID and InstructorName could not be null.

**Ratings**

This table uses CourseID and StudentID as its primary keys. There is a foreign key/RI constraint: each CourseID and StudentID in Ratings must correspond to an existing CourseID and StudentID in Courses and Students, respectively. If either is deleted in the Courses or Students tables, they will be deleted in the Ratings table as well.

**UnivCourses**

This table uses CourseID and UniversityID as its primary keys. There is a foreign key/RI constraint: each CourseID and UniversityID in UnivCourses must correspond to an existing CourseID and UniversityID in Courses and University, respectively. If either is deleted in the Courses or University tables, they will be deleted in the UnivCourses table as well.

**UnivInstructors**

This table uses InstructorID and UniversityID as its primary keys. There is a foreign key/RI constraint: each InstructorID and UniversityID in UnivInstructors must correspond to an existing InstructorID and UniversityID in Instructors and University, respectively. If either is deleted in the Instructors or University tables, they will be deleted in the UnivInstructors table as well.

**CourseInstructors**

This table uses InstructorID and CourseID as its primary keys. There is a foreign key/RI constraint: each InstructorID and CourseID in CourseInstructors must correspond to an existing InstructorID and CourseID in Instructors and Courses, respectively. If either is deleted in the Instructors or Courses tables, they will be deleted in the CourseInstructors table as well.

**StudentCourses**

This table uses StudentID and CourseID as its primary keys. There is a foreign key/RI constraint: each StudentID and CourseID in StudentCourses must correspond to an existing StudentID and CourseID in Students and Courses, respectively. If either is deleted in the Students or Courses tables, they will be deleted in the StudentCourses table as well.

**StudentCourseTrans**

This table uses StudentID, CourseID and TransactionID as its primary keys. There is a foreign key/RI constraint: each StudentID, CourseID and TransactionID in StudentCourseTrans must correspond to an existing StudentID, CourseID and TransactionID in Students, Courses and Transactions, respectively. If either is deleted in the Students, Courses or Transactions tables, they will be deleted in the StudentCourseTrans table as well.

**StudentCards**

This table uses StudentID and CreditCardNo as its primary keys. There is a foreign key/RI constraint: each StudentID and CreditCardNo in StudentCards must correspond to an existing StudentID and CreditCardNo in Students and CreditCards, respectively. If either is deleted in the Students or CreditCards tables, they will be deleted in the StudentCards table as well.

**CardTrans**

This table uses TransactionID and CreditCardNo as its primary keys. There is a foreign key/RI constraint: each TransactionID and CreditCardNo in CardTrans must correspond to an existing TransactionID and CreditCardID in Transactions and CreditCards, respectively. If either is deleted in the Transactions or CreditCards tables, they will be deleted in the CardTrans table as well.