Database Relational Model

Advisor Services Portal v1.1

The author of this program, Thomas Dye, may be reached through email at thomas@tfenet.com.

Table of Contents

Contents

Introduction

This document covers the database schema of the Advisor Portal website, as it exists in March of 2017. Each heading contains the following information:

- Table or stored procedure name
- Website application code dependencies
- Trigger dependencies
- Stored procedure dependencies
- A basic description or intended use with examples as appropriate
- An outline of the columns, keys, and constraints in use
- Raw T-SQL code

This document is structured so that tables are presented in an order which there are no foreign key constraints on tables not yet declared. Additionally, tables with listed website application code dependencies contain a screenshot clip of the primary UI element being rendered by a query using that table. If a dependency is listed but no screenshot exists, either no UI elements are directly rendered with that table or the entire page concentrates on that table and it us up to the developer to decide which is true. In all cases, the developer should consult the ColdFusion code for that page directly to interpret the table's usage as the screenshots are meant to act as a quick reference only.

Table naming in the database is meant to imply relationships between various groups of normalized tables. An example is that DEPARTMENT_RANKINGS is related to the DEPARTMENTS table, where the root name matches the primary table. A similar scheme is used with column naming where foreign key constraints exist. An example is [courses_id] where the foreign key is the column [id] in the table COURSES.

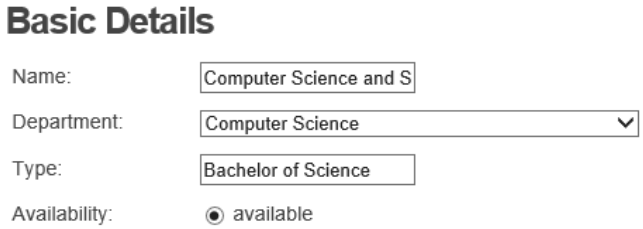
Foreign key cascading maintains database consistency with the exception of where multipathing may exist (and is disallowed in MS SQL) because of the complex topology of the relational model. In those events, triggers are used to ensure that INSERT, UPDATE, and DELETE operations do not cause data inconsistency. It is necessary to review the T-SQL code to understand where these limitations exist.

As a developer, you are encouraged to continue maintaining this document as the Advisor Portal project evolves, as having a central reference for schema documentation is valuable. Please treat this as a living document and make appropriate changes as they are committed to code.

DEPARTMENTS [Table]

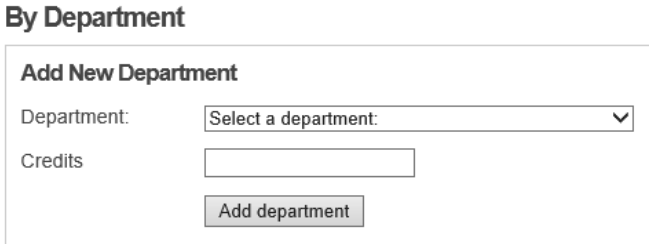
Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm

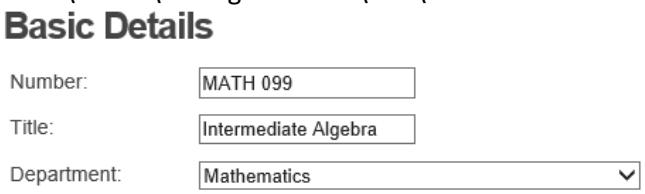


- AdvisorPortal\admin\manage-colleges\degrees\index.cfm

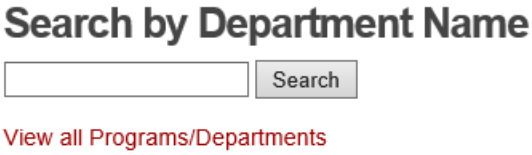
- AdvisorPortal\admin\manage-colleges\edit\index.cfm



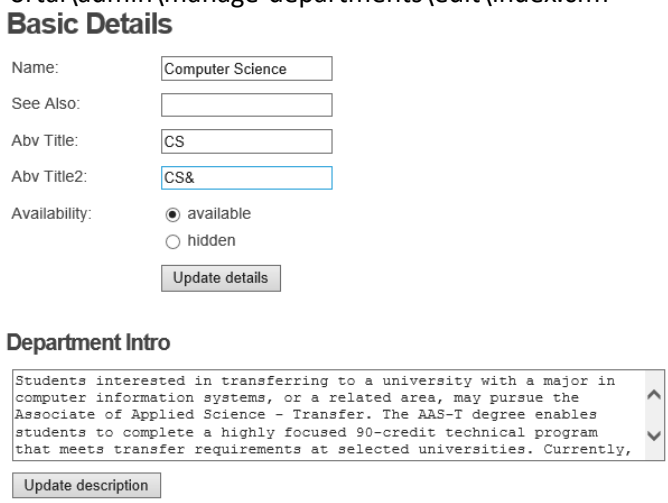
- AdvisorPortal\admin\manage-courses\edit\index.cfm



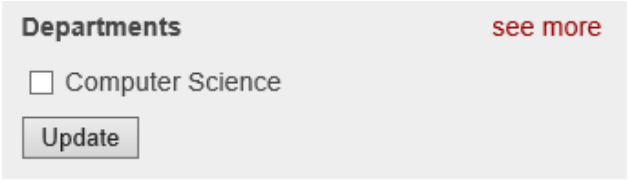
- AdvisorPortal\admin\manage-departments\index.cfm



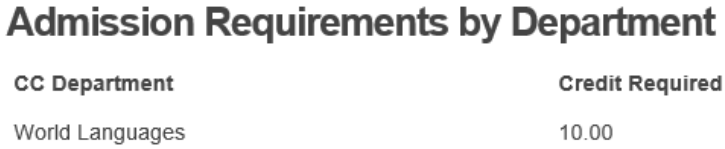
- AdvisorPortal\admin\manage-departments\edit\index.cfm



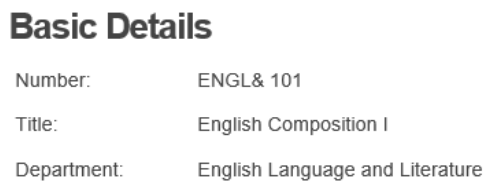
- AdvisorPortal\plans\create-plan\index.cfm



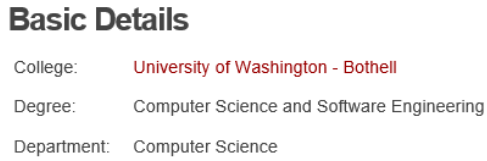
- AdvisorPortal\view\colleges\index.cfm



- AdvisorPortal\view\courses\index.cfm



- AdvisorPortal\view\degrees\index.cfm



This data is imported from the college. It is meant to mirror data already in use at EvCC and should not undergo much customization. The departments table contains the total number of departments at EvCC. Setting the use_column bit will allow departments to be available for use in the Advisor Portal.

Columns:

- id
  - INT
  - Not null
  - Primary key
- department_name
  - VARCHAR(255)
  - Not null
  - Unique
- see_also

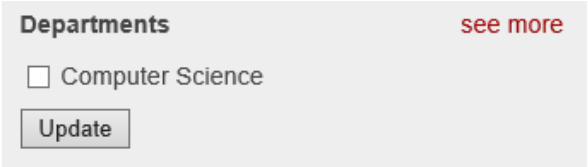
    - VARCHAR(255)
- dept_intro
    - VARCHAR(1800)
- abv_title
    - VARCHAR(10)
- abv_title2
    - VARCHAR(10)
- use_catalog
    - BIT
    - Not null

```
CREATE TABLE [dbo].[DEPARTMENTS] (
[id] INT NOT NULL,
[department_name] VARCHAR (255) NOT NULL,
[see_also] VARCHAR (255) NULL,
[dept_intro] VARCHAR (1800) NULL,
[abv_title] VARCHAR (10) NULL,
[abv_title2] VARCHAR (10) NULL,
[use_catalog] BIT NOT NULL,
CONSTRAINT [PK_DEPARTMENTS] PRIMARY KEY CLUSTERED ([id] ASC)
);
```

DEPARTMENT_RANKINGS [Table]

Used in the following pages:

- AdvisorPortal\plans\create-plan\index.cfm



Used in the following triggers:

- Trigger_PLAN_ACTIVEPLANS_insert
- Trigger_PLAN_ACTIVEPLANS_delete

Used to generate popularity lists for transfer universities when displaying degree search filters. Ranks are updated by triggers when students select schools for their degree plans.

Columns:

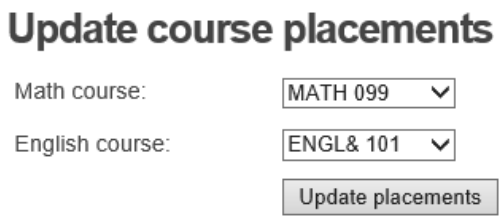
- departments_id
    - INT
    - Not null
    - Primary key
    - Foreign key (DEPARTMENTS id)
- rank
    - INT
    - Default 0
    - Constraint: rank >= 0

```sql
CREATE TABLE [dbo].[DEPARTMENT_RANKINGS] (
[departments_id] INT NOT NULL,
[rank] INT DEFAULT ((0)) NOT NULL,
CONSTRAINT [PK_DEPARTMENT_RANKINGS] PRIMARY KEY CLUSTERED ([departments_id] ASC),
CONSTRAINT [FK_DEPARTMENT_RANKINGS_departments_id] FOREIGN KEY ([departments_id])
REFERENCES [dbo].[DEPARTMENTS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [CK_DEPARTMENT_RANKINGS_rank] CHECK ([rank]>=(0))
);
```

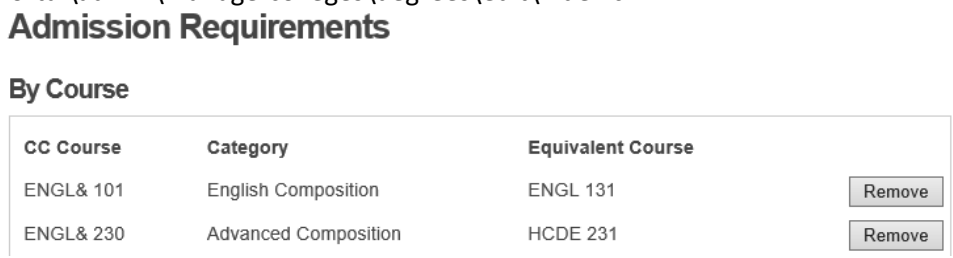
COURSES [Table]

Used in the following pages:

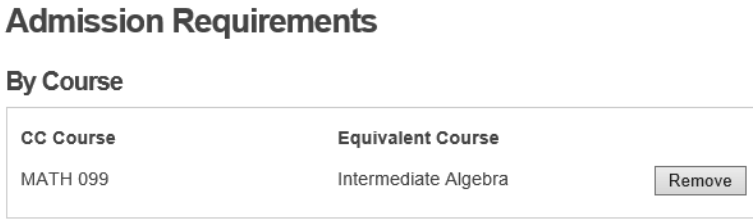
- AdvisorPortal\account\index.cfm



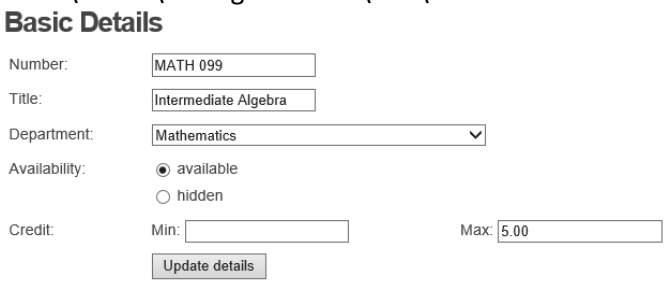
- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm



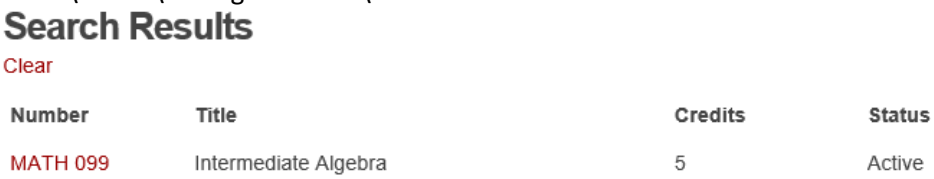
- AdvisorPortal\admin\manage-colleges\edit\index.cfm



- AdvisorPortal\admin\manage-courses\edit\index.cfm



- AdvisorPortal\admin\manage-courses\index.cfm



- AdvisorPortal\admin\manage-users\edit\index.cfm

Account

Email address:	student@email.com
First name:	Test
Last name:	Student
Student ID:	1234
Math course:	MATH 099
English course:	ENGL& 101

- AdvisorPortal\courses\index.cfm

Completed Courses

Number	Title	Credits	
ENGL& 101	English Composition I	5	Delete

- AdvisorPortal\dashboard\index.cfm

Courses remaining for this plan

English Composition

Code	Title	Credits	Status
ENGL& 101	English Composition I	5	

- AdvisorPortal\plans\edit\index.cfm

English Composition

Code	Title	Credits	Status	Remove
ENGL& 101	English Composition I	5		☐
				Update

- AdvisorPortal\view\colleges\index.cfm

Admission Requirements by Course

Minimum Admission Requirements All transfer applicants must meet the follow considered for admissions: 1. Minimum cumulative transfer GPA of 2.0. 2. Sup test scores, personal statement, etc. - varies by major program). 3. Completior requirements (see below). 4. Proof of English language proficiency (if required

CC Course	Equivalent Course
MATH 099	Intermediate Algebra

- AdvisorPortal\view\courses\index.cfm

Basic Details

Number:	MATH 099
Title:	Intermediate Algebra
Department:	Mathematics
Credit:	5.00

- AdvisorPortal\view\degrees\index.cfm

Admission Requirements by Course

The table below lists the prerequisite courses and other requirements for the Computer Science an Engineering (CSSE) majors offered by the CSS Division. Because the UW course equivalency guid given using UW Seattle course numbers, those are provided in the second column. All of the above requirements for the major you are applying to must be completed prior to applying to the major. Th is Statistics which is the ONLY soft prerequisite; this means that course can be in progress or comp being admitted to the major. Please note that both Programming I and II must be taken in the same programming language. There is no requirement as to which programming language is used as lon courses are taught using the same programming language.

CC Course	Category	Equivalent Course
ENGL& 101	English Composition	ENGL 131

Used in the following triggers:

- Trigger_STUDENTS_COMPLETEDCOURSES_insert
- Trigger_STUDENTS_COMPLETEDCOURSES_delete

Used in the following stored procedures:

- createPLANS

This data is imported from the college. It is meant to mirror data already in use at EvCC and should not undergo much customization. Like departments, courses have a use_catalog bit to enable/disable them. Courses are associated with departments and cannot exist without one. As a result, they are subject to deactivation if the department use_catalog is set disabled, regardless of the course use_catalog bit. A modification of the original EvCC data was that credits were divided into MinCredit and MaxCredit since variable credit courses were stored as a range string (1-5).

- id
  - INT
  - Not null
  - Primary key
- course_number
  - VARCHAR(13)
  - Not null
- title
  - VARCHAR(255)
  - Not null
- min_credit
  - DECIMAL(4,2)
  - Constraint: min_credit >= 0
- max_credit
  - DECIMAL(4,2)
  - Not null
  - Constraint: IF (min_credit IS NOT NULL) THEN max_credit >= min_credit
- course_description
  - VARCHAR(1800)
- departments_id
  - INT
  - Not null
  - Foreign key (DEPARTMENTS id)
- use_catalog
  - BIT
  - Not null

```
CREATE TABLE [dbo].[COURSES] (
[id] INT NOT NULL,
[course_number] VARCHAR (13) NOT NULL,
[title] VARCHAR (255) NOT NULL,
[min_credit] DECIMAL (4, 2) NULL,
[max_credit] DECIMAL (4, 2) NOT NULL,
[course_description] VARCHAR (1800) NULL,
```

```
[departments_id] INT NOT NULL,
[use_catalog] BIT NOT NULL,
CONSTRAINT [PK_COURSES] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_COURSES_departments_id] FOREIGN KEY ([departments_id]) REFERENCES
[dbo].[DEPARTMENTS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [CK_COURSES_max_credit] CHECK (([min_credit] IS NULL OR
[max_credit]>=[min_credit]) AND [max_credit]>=(0)),
CONSTRAINT [CK_COURSES_min_credit] CHECK ([min_credit]>=(0))
);
```

COURSE_RANKINGS [Table]

Used in the following pages:

- N/A

Used in the following triggers:

- Trigger_STUDENTS_COMPLETEDCOURSES_insert
- Trigger_STUDENTS_COMPLETEDCOURSES_delete

Tracks how many of which course has been completed by students. As students add courses to the Completed Courses page, these ranks are updated by triggers.

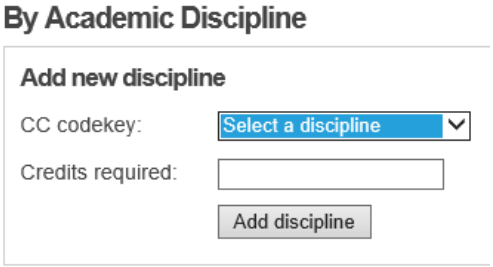
- courses_id
  - INT
  - Not null
  - Primary key
  - Foreign key (COURSES id)
- rank
  - INT
  - Default 0
  - Constraint: Rank >= 0

```
CREATE TABLE [dbo].[COURSE_RANKINGS] (
[courses_id] INT NOT NULL,
[rank] INT DEFAULT ((0)) NOT NULL,
CONSTRAINT [PK_COURSE_RANKINGS] PRIMARY KEY CLUSTERED ([courses_id] ASC),
CONSTRAINT [FK_COURSE_RANKINGS_courses_id] FOREIGN KEY ([courses_id]) REFERENCES
[dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [CK_COURSE_RANKINGS_rank] CHECK ([rank]>=(0))
);
```

CODEKEYS [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\edit\index.cfm



- AdvisorPortal\view\colleges\index.cfm

Admission Requirements by Academic Discipline

EvCC Codekey	Credit Required
Diversity	3.00

Defines an arbitrary list of course classifications codes as decided by EvCC in the course catalog. This allows identification of certain types of courses (humanities, diversity, etc) through query. Example: AAS, C, NS, H, SS, NS-L, Q, TE, D

- id
  - INT
  - Auto increment
  - Not null
  - Primary key
- codekey
  - VARCHAR(4)
  - Not null
- description
  - VARCHAR(50)
  - Not null

```
CREATE TABLE [dbo].[CODEKEYS] (
[id] INT IDENTITY (1, 1) NOT NULL,
[codekey] VARCHAR (4) NOT NULL,
[description] VARCHAR (50) NOT NULL,
CONSTRAINT [PK_CODEKEYS] PRIMARY KEY CLUSTERED ([id] ASC)
);
```

COURSES_CODEKEYS [Table]

Used in the following pages:

- N/A

Creates a mapping of courses to codekeys. Some classes have multiple codekeys and can be classified in various ways, depending on instruction requirement.

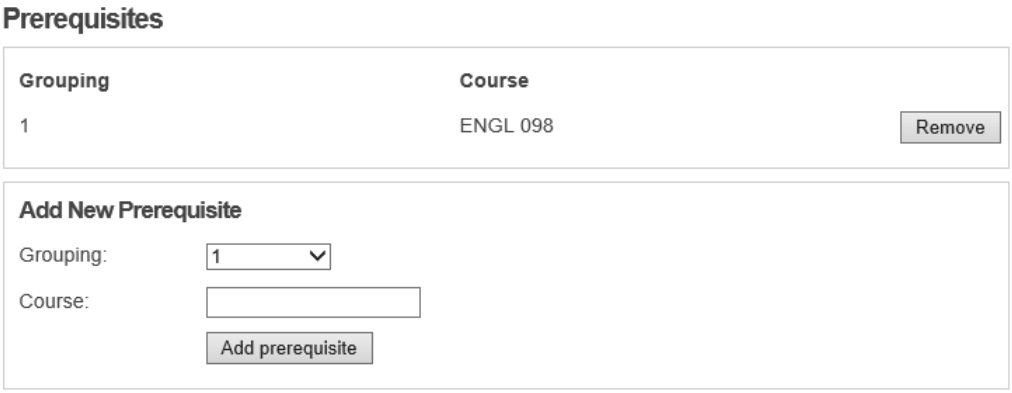
- courses_id
  - INT
  - Not null
  - Foreign key (COURSES id)
- codekeys_id
  - INT
  - Not null
  - Primary key (courses_id, codekeys_id)
  - Foreign key (CODEKEYS id)

```
CREATE TABLE [dbo].[COURSES_CODEKEYS] (
[courses_id] INT NOT NULL,
[codekeys_id] INT NOT NULL,
CONSTRAINT [PK_COURSES_CODEKEYS] PRIMARY KEY CLUSTERED ([codekeys_id] ASC,
[courses_id] ASC),
CONSTRAINT [FK_COURSES_CODEKEYS_codekeys_id] FOREIGN KEY ([codekeys_id])
REFERENCES [dbo].[CODEKEYS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_COURSES_CODEKEYS_courses_id] FOREIGN KEY ([courses_id]) REFERENCES
[dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

PREREQUISITES [Table]

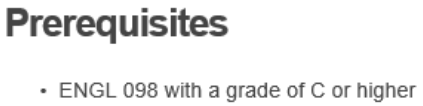
Used in the following pages:

- AdvisorPortal\admin\manage-courses\edit\index.cfm



- AdvisorPortal\courses\index.cfm

- AdvisorPortal\view\courses\index.cfm



Create a mapping of course prerequisite requirements. For any course, it may have another course that must be completed prior to taking said course. This table fulfils that mapping. It only defines course prerequisites at the community college and does not refer to course prerequisites for degrees at universities.

Simplified prerequisites table. Example: BIOL& 231

Prerequisites string as found in CIF: BIOL& 211, or BIOL& 221 and BIOL& 222; and CHEM& 121 or CHEM&161 plus 162 or higher all with a grade of C or higher; or instructor permission.

The course logical equivalent structure is (BIOL&211 OR (BIOL&221 AND BIOL&222)) AND (CHEM&121 OR (CHEM&161 AND CHEM&162))

In a database, this can be broken down into the following individual groupings:

* Instructor permission.
* BIOL& 211 and CHEM& 121
* BIOL& 211 and CHEM& 161 and CHEM& 162
* BIOL& 221 and BIOL& 222 and CHEM& 121
* BIOL& 221 and BIOL& 222 and CHEM& 161 and CHEM& 162

All element in a grouping are considered to have an AND relationship. Different groupings for the same course are considered to have an OR relationship. Each individual grouping can be parsed, where all elements of group A are enough to meet prerequisites, OR all elements of group B are enough to meet prerequisites, OR group C, etc. Satisfying one grouping is sufficient. It is not necessary to continue parsing after the entirety of a single group is satisfied.

- id
  - INT
  - Auto increment
  - Not null
  - Primary key
- courses_id

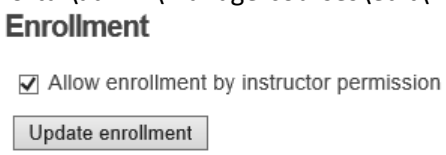
    - INT
    - Not null
    - Foreign key (COURSES id)
- group_id
    - INT
    - Not null
- courses_prerequisite_id
    - INT
    - Not null
    - Foreign key (COURSES id)
    - Constraint: courses_prerequisite_id IS NOT courses_id

```
CREATE TABLE [dbo].[PREREQUISITES] (
[id] INT IDENTITY (1, 1) NOT NULL,
[courses_id] INT NOT NULL,
[group_id] INT NOT NULL,
[courses_prerequisite_id] INT NOT NULL,
CONSTRAINT [PK_PREREQUISITES] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_PREREQUISITES_courses_id] FOREIGN KEY ([courses_id]) REFERENCES
[dbo].[COURSES] ([id]),
CONSTRAINT [FK_PREREQUISITES_courses_prequiresite_id] FOREIGN KEY
([courses_prerequisite_id]) REFERENCES [dbo].[COURSES] ([id]),
CONSTRAINT [CK_PREREQUISITES_courses_prerequisite_id] CHECK
([courses_id]<>[courses_prerequisite_id])
);
```

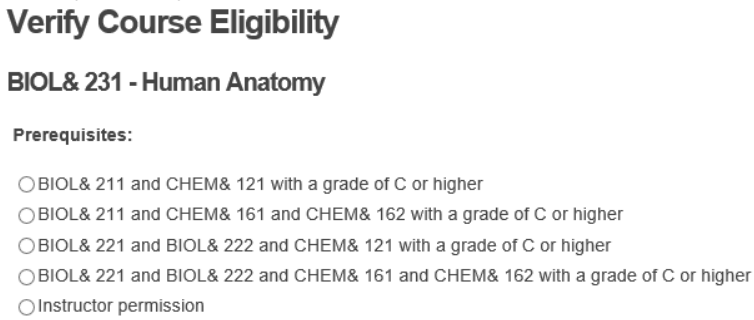
PREREQUISITE_PERMISSIONS [Table]

Used in the following pages:

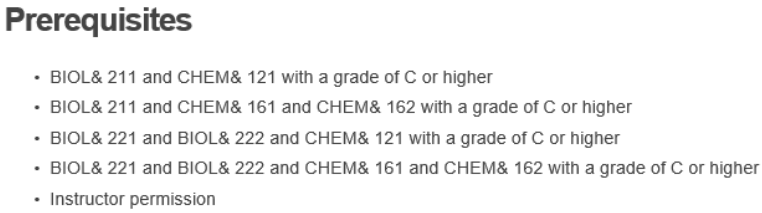
- AdvisorPortal\admin\manage-courses\edit\index.cfm



- AdvisorPortal\courses\index.cfm



- AdvisorPortal\view\courses\index.cfm



Specify if instructor permission is allowed as a prerequisite option for course enrollment. This data is informational in nature. This provides an alternate, manual option for students self-certifying that they were qualified for entering in a

completed course.

- courses_id
  - INT
  - Not null
  - Primary key
  - Foreign key (COURSES id)

```
CREATE TABLE [dbo].[PREREQUISITE_PERMISSIONS] (
[courses_id] INT NOT NULL,
CONSTRAINT [PK_PREREQUISITE_PERMISSIONS] PRIMARY KEY CLUSTERED ([courses_id] ASC),
CONSTRAINT [FK_PREREQUISITE_PERMISSIONS_courses_id] FOREIGN KEY ([courses_id])
REFERENCES [dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

PREREQUISITE_PLACEMENTS [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-courses\edit\index.cfm

  Placement Scores

  82 or higher on Compass Test [Remove]

- AdvisorPortal\courses\index.cfm

  Verify Course Eligibility

  ENGL& 101 - English Composition I

  Prerequisites:

  ○ ENGL 098 with a grade of C or higher
  ○ Placement into ENGL& 101 by assessment.

- AdvisorPortal\view\courses\index.cfm

  Prerequisites

  - ENGL 098 with a grade of C or higher
  - Placement into ENGL& 101 by assessment.

Satisfies course requirement where a placement exam qualifies a student to be in a single specific course. This data is informational in nature. Example: Required “82 or higher on Compass test”. This provides an alternate, manual option for students self-certifying that they were qualified for entering in a completed course.

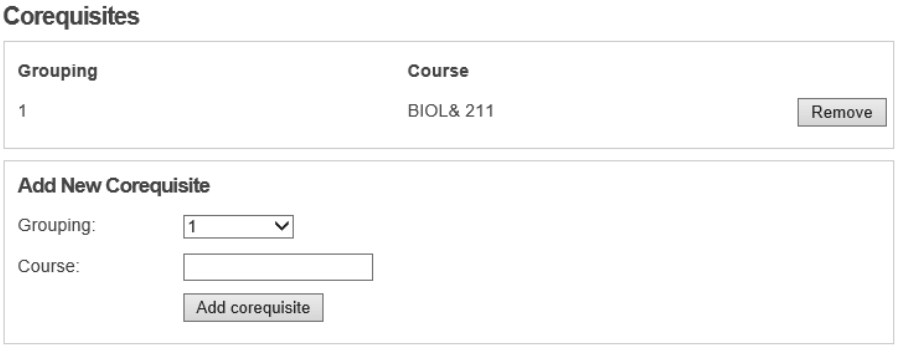
- courses_id
  - INT
  - Not null
  - Primary key
  - Foreign key (COURSES id)
- placement
  - VARCHAR(255)
  - Not null

```
CREATE TABLE [dbo].[PREREQUISITE_PLACEMENTS] (
[courses_id] INT NOT NULL,
[placement] VARCHAR (255) NOT NULL,
CONSTRAINT [PK_PREREQUISITE_PLACEMENTS] PRIMARY KEY CLUSTERED ([courses_id] ASC),
CONSTRAINT [FK_PREREQUISITE_PLACEMENTS_courses_id] FOREIGN KEY ([courses_id])
REFERENCES [dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

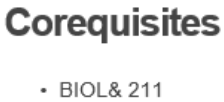
COREQUISITES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-courses\edit\index.cfm



- AdvisorPortal\view\courses\index.cfm



See prerequisite entity for notes. This structure is nearly identical. This table creates a course mapping for the recommendation engine to determine schedule planning, specifically where some courses must be taken either before, or at the same time, of a given course. Example: Course B must be completed before taking course A, except where a student is concurrently enrolled in both course A and course B.

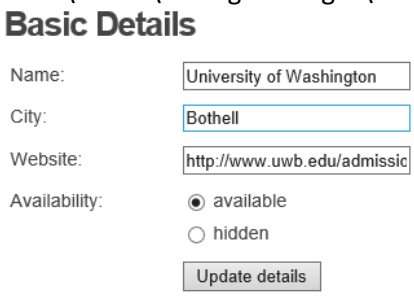
- id
  - INT
  - Auto increment
  - Not null
  - Primary key
- courses_id
  - INT
  - Not null
  - Foreign key (COURSES id)
- group_id
  - INT
  - Not null
- courses_corequisite_id
  - INT
  - Not null
  - Foreign key (COURSES id)
  - Constraint: courses_corequisite_id IS NOT courses_id

```
CREATE TABLE [dbo].[COREQUISITES] (
[id] INT IDENTITY (1, 1) NOT NULL,
[courses_id] INT NOT NULL,
[group_id] INT NOT NULL,
[courses_corequisite_id] INT NOT NULL,
CONSTRAINT [PK_COREQUISITES] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_COREQUISITES_courses_id] FOREIGN KEY ([courses_id]) REFERENCES
[dbo].[COURSES] ([id]),
CONSTRAINT [FK_COREQUISITES_courses_corequisite_id] FOREIGN KEY
([courses_corequisite_id]) REFERENCES [dbo].[COURSES] ([id]),
CONSTRAINT [CK_PREREQUISITES_courses_corequisite_id] CHECK
([courses_id]<>[courses_corequisite_id])
);
```

COLLEGES [Table]

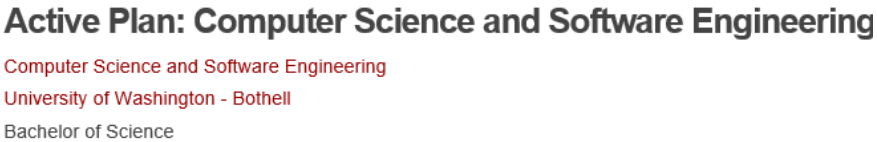
Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm
- AdvisorPortal\admin\manage-colleges\degrees\index.cfm
- AdvisorPortal\admin\manage-colleges\edit\index.cfm



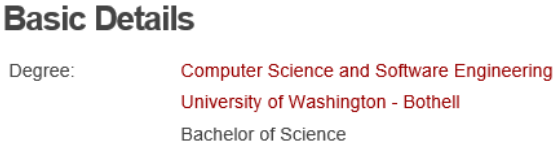
- AdvisorPortal\admin\manage-colleges\index.cfm

- AdvisorPortal\dashboard\index.cfm

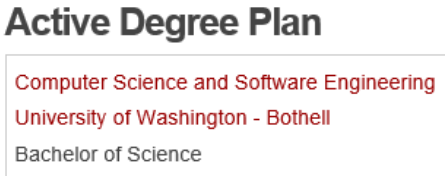


- AdvisorPortal\plans\create-plan\index.cfm

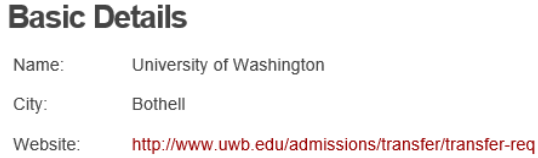
- AdvisorPortal\plans\edit\index.cfm



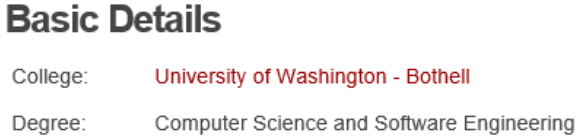
- AdvisorPortal\plans\index.cfm



- AdvisorPortal\view\colleges\index.cfm



- AdvisorPortal\view\degrees\index.cfm



Used in the following stored procedures:

- createPLANS

Defines the list of tracked colleges available for transfer from EvCC and populates the filter list on degree search. The use_catalog bit may be used to delist a college until it is ready for public viewing. A delisted college is available to editors and administrators only, but may not be used in creating new degree plans.

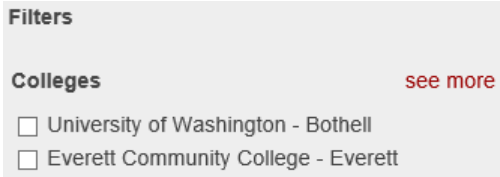
- id
  - INT
  - Auto increment
  - Not null
  - Primary key
- college_name
  - VARCHAR(255)
  - Not null
- college_city
  - VARCHAR(255)
  - Not null
- college_website
  - VARCHAR(255)
- use_catalog
  - Boolean
  - Not null

```
CREATE TABLE [dbo].[COLLEGES] (
[id] INT IDENTITY (1, 1) NOT NULL,
[college_name] VARCHAR (255) NOT NULL,
[college_city] VARCHAR (255) NOT NULL,
[college_website] VARCHAR (255) NULL,
[use_catalog] BIT NOT NULL,
CONSTRAINT [PK_COLLEGES] PRIMARY KEY CLUSTERED ([id] ASC)
);

GO
-- Thomas Dye, August 2016 --
CREATE TRIGGER [dbo].[Trigger_COLLEGES_insert]
ON [dbo].[COLLEGES]
FOR INSERT
AS
BEGIN
SET NoCount ON
             -- Ensure that a location to save notes exists for this college
             DECLARE @College_ID INT
             SET @College_ID = (SELECT id FROM inserted)
             IF NOT EXISTS (SELECT colleges_id FROM COLLEGE_NOTES WHERE
colleges_id = @College_ID)
                   INSERT INTO COLLEGE_NOTES (colleges_id) VALUES (@College_ID)
END
```

COLLEGE_RANKINGS [Table]

Used in the following pages:

- AdvisorPortal\plans\create-plan\index.cfm



Used in the following triggers:

- Trigger_PLAN_ACTIVEPLANS_insert
- Trigger_PLAN_ACTIVEPLANS_delete

Used to generate popularity lists for transfer universities when displaying degree search filters. Ranks are updated by triggers when students select a school for their active degree plan.

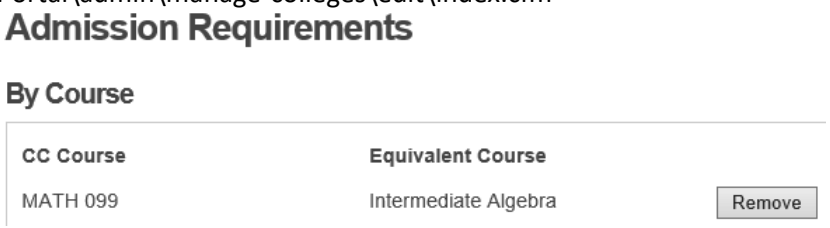
- colleges_id
  - INT
  - Not null
  - Primary key
  - Foreign key (COLLEGES id)
- rank
  - INT
  - Default 0
  - Constraint: Rank >= 0

```
CREATE TABLE [dbo].[COLLEGE_RANKINGS] (
[colleges_id] INT NOT NULL,
[rank] INT DEFAULT ((0)) NOT NULL,
CONSTRAINT [PK_COLLEGE_RANKINGS] PRIMARY KEY CLUSTERED ([colleges_id] ASC),
CONSTRAINT [FK_COLLEGE_RANKINGS_colleges_id] FOREIGN KEY ([colleges_id])
REFERENCES [dbo].[COLLEGES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [CK_COLLEGE_RANKINGS_rank] CHECK ([rank]>=(0))
);
```

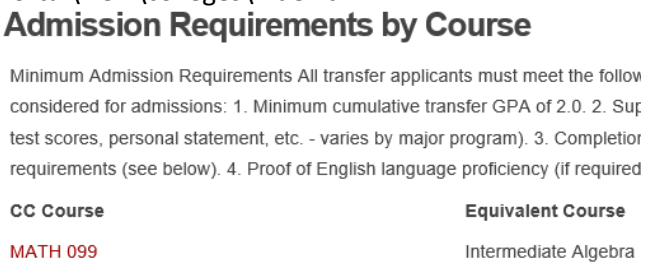
COLLEGE_ADMISSION_COURSES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\edit\index.cfm



- AdvisorPortal\view\colleges\index.cfm



Used in the following stored procedures:

- createPLANS

Use when the admission requirements of a college call out specific courses that need to be taken to be taken by a student prior to college enrollment. The courses_id column is the CC equivalency for the university specified course where the foreign_course_number is a label identifying the name of the university course for informational purposes

only. Each college may only identify a courses_id once.

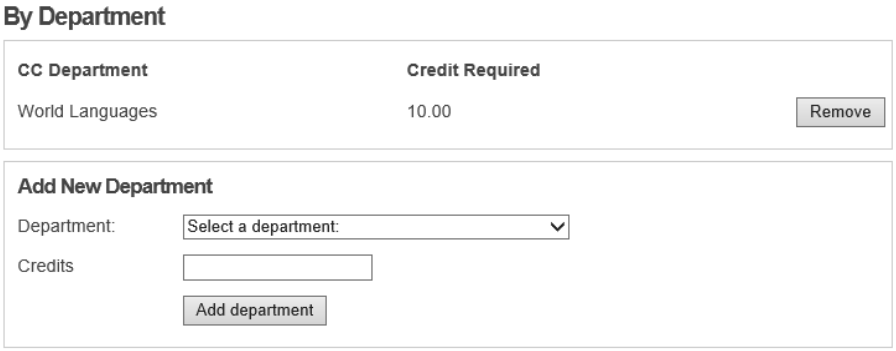
- colleges_id
  - INT
  - Not null
  - Foreign key (COLLEGES id)
- courses_id
  - INT
  - Not null
  - Primary key (colleges_id, courses_id)
  - Foreign key (COURSES id)
- foreign_course_number
  - VARCHAR(50)
  - Not null

```
CREATE TABLE [dbo].[COLLEGE_ADMISSION_COURSES] (
[colleges_id] INT NOT NULL,
[courses_id] INT NOT NULL,
[foreign_course_number] VARCHAR (50) NOT NULL,
CONSTRAINT [PK_COLLEGES_ADMISSION_COURSES] PRIMARY KEY CLUSTERED ([colleges_id]
ASC, [courses_id] ASC),
CONSTRAINT [FK_COLLEGES_ADMISSION_COURSES_colleges_id] FOREIGN KEY ([colleges_id])
REFERENCES [dbo].[COLLEGES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_COLLEGES_ADMISSION_COURSES_courses_id] FOREIGN KEY ([courses_id])
REFERENCES [dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

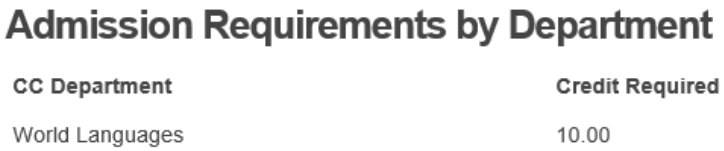
COLLEGE_ADMISSION_DEPARTMENTS [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\edit\index.cfm



- AdvisorPortal\view\colleges\index.cfm



Use when the admission requirements of a college call out a category of courses that need to be taken. Example: college requires 10 credits of world languages for admission.

- colleges_id
  - INT
  - Not null
  - Foreign key (COLLEGES id)
- departments_id

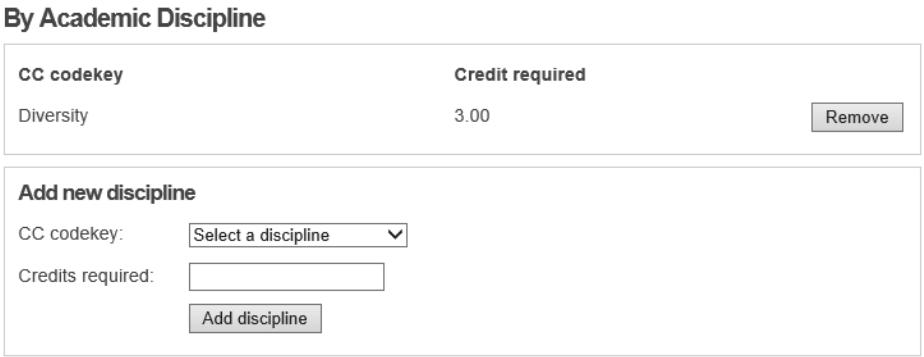
    - INT
    - Not null
    - Primary key (colleges_id, departments_id)
    - Foreign key (DEPARTMENTS id)
- credit
    - DECIMAL(4,2)
    - Not null

```
CREATE TABLE [dbo].[COLLEGE_ADMISSION_DEPARTMENTS] (
[colleges_id] INT NOT NULL,
[departments_id] INT NOT NULL,
[credit] DECIMAL (4, 2) NOT NULL,
CONSTRAINT [PK_COLLEGE_ADMISSION_DEPARTMENTS] PRIMARY KEY CLUSTERED ([colleges_id]
ASC, [departments_id] ASC),
CONSTRAINT [FK_COLLEGE_ADMISSION_DEPARTMENTS_colleges_id] FOREIGN KEY
([colleges_id]) REFERENCES [dbo].[COLLEGES] ([id]) ON DELETE CASCADE ON UPDATE
CASCADE,
CONSTRAINT [FK_COLLEGE_ADMISSION_DEPARTMENTS_departments_id] FOREIGN KEY
([departments_id]) REFERENCES [dbo].[DEPARTMENTS] ([id]) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

COLLEGE_ADMISSION_CODEKEYS [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\edit\index.cfm



- AdvisorPortal\view\colleges\index.cfm

Use when the admission requirements of a college call out types of courses that need to be taken. Example: college requires 5 credits of social sciences.

- colleges_id
    - INT
    - Not null
    - Foreign key (COLLEGES id)
- codekeys_id
    - INT
    - Not null
    - Primary key (colleges_id, codekeys_id)
    - Foreign key (CODEKEYS id)
- credit

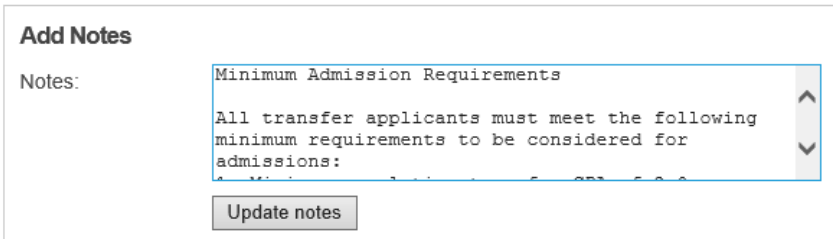
- DECIMAL(4,2)
- Not null

```
CREATE TABLE [dbo].[COLLEGE_ADMISSION_CODEKEYS] (
[colleges_id] INT NOT NULL,
[codekeys_id] INT NOT NULL,
[credit] DECIMAL (4, 2) NOT NULL,
CONSTRAINT [PK_COLLEGE_ADMISSION_CODEKEYS] PRIMARY KEY CLUSTERED ([colleges_id]
ASC, [codekeys_id] ASC),
CONSTRAINT [FK_COLLEGE_ADMISSION_CODEKEYS_codekeys_id] FOREIGN KEY ([codekeys_id])
REFERENCES [dbo].[CODEKEYS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_COLLEGE_ADMISSION_CODEKEYS_colleges_id] FOREIGN KEY ([colleges_id])
REFERENCES [dbo].[COLLEGES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

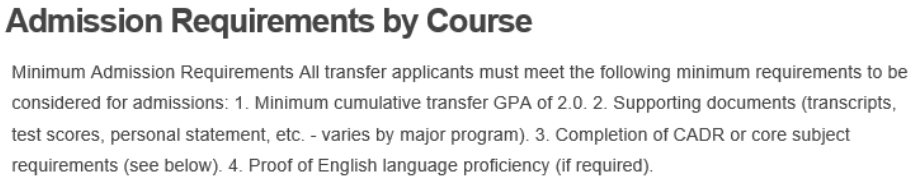
COLLEGE_NOTES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\edit\index.cfm



- AdvisorPortal\view\colleges\index.cfm



Used in the following triggers:

- Trigger_COLLEGES_insert

Store notes relating to college enrollment that are viewable to advisors and students. For any college, notes may exist for required admission courses, department (see COLLEGE_ADMISSION_DEPARTMENTS World Languages example), and/or codekeys (see COLLEGE_ADMISSION_CODEKEYS Social Sciences example).

- colleges_id
  - INT
  - Not null
  - Primary key
  - Foreign key (COLLEGES id)
- courses_note
  - VARCHAR(1800)
- departments_note
  - VARCHAR(1800)
- codekeys_note
  - VARCHAR(1800)

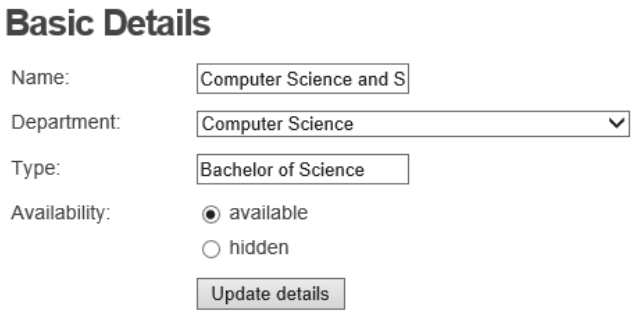
```
CREATE TABLE [dbo].[COLLEGE_NOTES] (
[colleges_id] INT NOT NULL,
[courses_note] VARCHAR (1800) NULL,
[departments_note] VARCHAR (1800) NULL,
[codekeys_note] VARCHAR (1800) NULL,
```

```
CONSTRAINT [PK_COLLEGE_NOTES] PRIMARY KEY CLUSTERED ([colleges_id] ASC),
CONSTRAINT [FK_COLLEGE_NOTES_colleges_id] FOREIGN KEY ([colleges_id]) REFERENCES
[dbo].[COLLEGES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

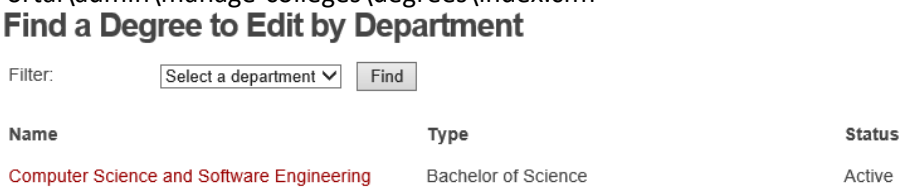
DEGREES [Table]

Used in the following pages:

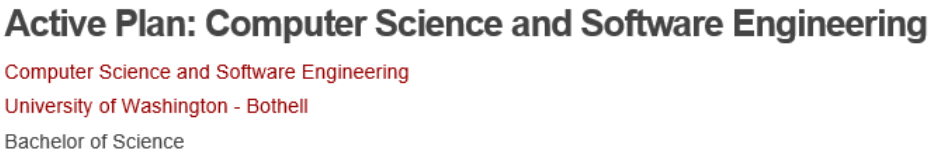
- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm



- AdvisorPortal\admin\manage-colleges\degrees\index.cfm

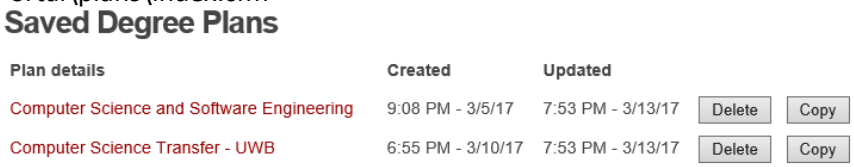


- AdvisorPortal\dashboard\index.cfm



- AdvisorPortal\plans\create-plan\index.cfm

- AdvisorPortal\plans\edit\index.cfm
- AdvisorPortal\plans\index.cfm



- AdvisorPortal\view\degrees\index.cfm

Used in the following triggers:

- Trigger_PLAN_ACTIVEPLANS_insert
- Trigger_PLAN_ACTIVEPLANS_delete

Used in the following stored procedures:

- createPLANS

Defines the list of tracked degrees available for transfer from EvCC and populates the search results during degree search. The use_catalog bit may be used to delist a degree until it is ready for public viewing. A delisted degree is available to editors and administrators only, but may not be used in creating new degree plans.

- id

    - INT
    - Auto increment
    - Not null
    - Primary key
- colleges_id
    - INT
    - Not null
    - Foreign key (COLLEGES id)
- degree_name
    - VARCHAR(255)
    - Not null
- departments_id
    - INT
    - Not null
    - Foreign key (DEPARTMENTS id)
- degree_type
    - VARCHAR(255)
    - Not null
- use_catalog
    - Boolean
    - Not null

```
CREATE TABLE [dbo].[DEGREES] (
[id] INT IDENTITY (1, 1) NOT NULL,
[colleges_id] INT NOT NULL,
[degree_name] VARCHAR (255) NOT NULL,
[departments_id] INT NOT NULL,
[degree_type] VARCHAR (255) NOT NULL,
[use_catalog] BIT NOT NULL,
CONSTRAINT [PK_DEGREES] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_DEGREES_colleges_id] FOREIGN KEY ([colleges_id]) REFERENCES
[dbo].[COLLEGES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_DEGREES_departments_id] FOREIGN KEY ([departments_id]) REFERENCES
[dbo].[DEPARTMENTS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);

GO
-- Thomas Dye, August 2016 --
CREATE TRIGGER [dbo].[Trigger_DEGREES_insert]
ON [dbo].[DEGREES]
FOR INSERT
AS
BEGIN
SET NoCount ON
             -- Ensure that a location to save notes exists for this degree
             DECLARE @Degree_ID INT
             SET @Degree_ID = (SELECT id FROM inserted)
             IF NOT EXISTS (SELECT degrees_id FROM DEGREE_NOTES WHERE degrees_id =
@Degree_ID)
                    INSERT INTO DEGREE_NOTES (degrees_id) VALUES (@Degree_ID)
END
```

DEGREE_CATEGORIES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm

Degree Categories

Category	
English Composition	Remove
Advanced Composition	Remove
Mathematics	Remove

- AdvisorPortal\dashboard\index.cfm

Courses remaining for this plan

College Admission Courses

Code	Title	Credits	Status
MATH 099	Intermediate Algebra	5.00	Complete

- AdvisorPortal\plans\edit\index.cfm

College Admission Courses

Code	Title	Credits	Status	Remove
MATH 099	Intermediate Algebra	5.00	Complete	☐
				Update

- AdvisorPortal\view\degrees\index.cfm

Optional Graduation Requirements by Degree Category

CC Codekey	Credit Required
Visual, Literary, and Performing Arts	15.00
Individuals and Societies	15.00
Free Electives	25.00

Used in the following stored procedures:

- createPLANS

Defines an arbitrary list of categories for courses associated with a degree. This allows editors and administrators to create degrees with courses sorted into category names of their choosing. Example categories: Basic communication skills, Basic quantitative skills, Humanities

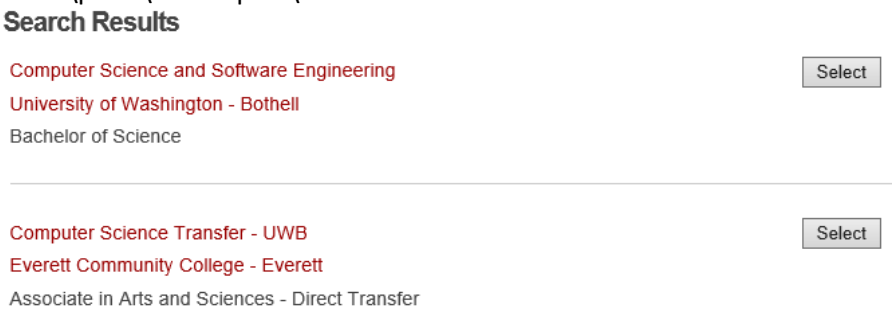
- id
  - INT
  - Auto increment
  - Not null
  - Primary key
- degrees_id
  - INT
  - Not null
  - Foreign key (DEGREES id)
- category
  - VARCHAR(50)
  - Not null

```
CREATE TABLE [dbo].[DEGREE_CATEGORIES] (
[id] INT IDENTITY (1, 1) NOT NULL,
[degrees_id] INT NOT NULL,
[category] VARCHAR (50) NOT NULL,
CONSTRAINT [PK_DEGREE_CATEGORIES] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_DEGREE_CATEGORIES_degrees_id] FOREIGN KEY ([degrees_id]) REFERENCES
[dbo].[DEGREES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

DEGREE_RANKINGS [Table]

Used in the following pages:

- AdvisorPortal\plans\create-plan\index.cfm



Used in the following triggers:

- Trigger_PLAN_ACTIVEPLANS_insert
- Trigger_PLAN_ACTIVEPLANS_delete

Used to generate ordering of search results when creating a new degree plan. Ranks are updated by triggers when students select a school for their active degree plan.

- degrees_id
  - INT
  - Not null
  - Primary key
  - Foreign key (DEGREES id)
- rank
  - INT
  - Default 0
  - Not null
  - Constraint: Rank >= 0

```
CREATE TABLE [dbo].[DEGREE_RANKINGS] (
[degrees_id] INT NOT NULL,
[rank] INT DEFAULT ((0)) NOT NULL,
CONSTRAINT [PK_DEGREE_RANKINGS] PRIMARY KEY CLUSTERED ([degrees_id] ASC),
CONSTRAINT [FK_DEGREE_RANKINGS_degrees_id] FOREIGN KEY ([degrees_id]) REFERENCES
[dbo].[DEGREES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [CK_DEGREE_RANKINGS_rank] CHECK ([rank]>=(0))
);
```

DEGREE_ADMISSION_COURSES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm

Admission Requirements

By Course

CC Course	Category	Equivalent Course	
ENGL& 101	English Composition	ENGL 131	Remove
ENGL& 230	Advanced Composition	HCDE 231	Remove

- AdvisorPortal\view\degrees\index.cfm

Admission Requirements by Course

The table below lists the prerequisite courses and other requirements for the Computer Science an Engineering (CSSE) majors offered by the CSS Division. Because the UW course equivalency guid given using UW Seattle course numbers, those are provided in the second column. All of the above requirements for the major you are applying to must be completed prior to applying to the major. Th is Statistics which is the ONLY soft prerequisite; this means that course can be in progress or comp being admitted to the major. Please note that both Programming I and II must be taken in the same programming language. There is no requirement as to which programming language is used as lon courses are taught using the same programming language.

CC Course	Category	Equivalent Course
ENGL& 101	English Composition	ENGL 131
ENGL& 230	Advanced Composition	HCDE 231

Used in the following stored procedures:

- createPLANS

Use when the admission requirements of a degree call out specific courses that need to be taken to be taken by a student prior to degree admission. The courses_id column is the CC equivalency for the university specified course where the foreign_course_number is a label identifying the name of the university course for informational purposes only. Each college may only identify a courses_id once.

- degrees_id
  - INT
  - Not null
  - Foreign key (DEGREES id)
- courses_id
  - INT
  - Not null
  - Primary key (degrees_id, courses_id)
  - Foreign key (COURSES id)
- degree_categories_id
  - INT
  - Not null
  - Foreign key (DEGREE_CATEGORIES id)
- foreign_course_number
  - VARCHAR(50)
  - Not null

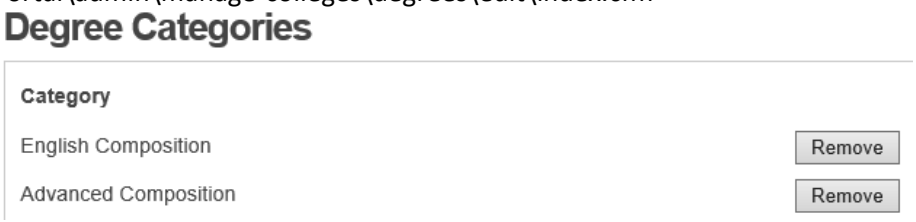
```
CREATE TABLE [dbo].[DEGREE_ADMISSION_COURSES] (
[degrees_id] INT NOT NULL,
[courses_id] INT NOT NULL,
[degree_categories_id] INT NOT NULL,
[foreign_course_number] VARCHAR (50) NOT NULL,
CONSTRAINT [PK_DEGREE_ADMISSION_COURSES] PRIMARY KEY CLUSTERED ([degrees_id] ASC,
[courses_id] ASC),
CONSTRAINT [FK_DEGREE_ADMISSION_COURSES_courses_id] FOREIGN KEY ([courses_id])
REFERENCES [dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
CONSTRAINT [FK_DEGREE_ADMISSION_COURSES_degrees_id] FOREIGN KEY ([degrees_id])
REFERENCES [dbo].[DEGREES] ([id]),
CONSTRAINT [FK_DEGREE_ADMISSION_COURSES_degree_categories_id] FOREIGN KEY
([degree_categories_id]) REFERENCES [dbo].[DEGREE_CATEGORIES] ([id])
);
```

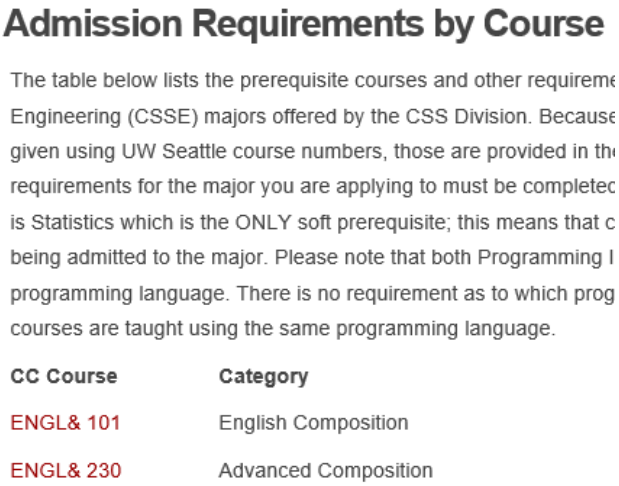
DEGREE_ADMISSION_CATEGORIES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm



- AdvisorPortal\view\degrees\index.cfm



Used to track when a degree requires a specified number of course credits to be taken in a user-defined degree category. Example: 5 VLPA credits, 5 I&S credits

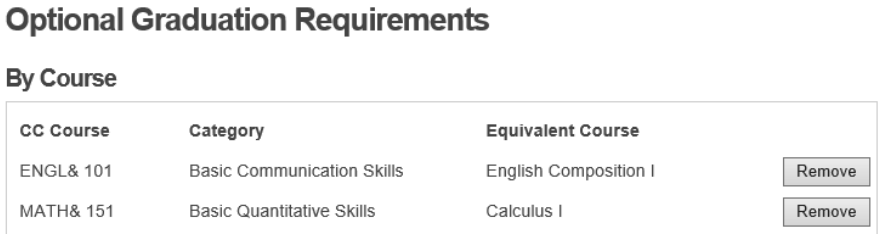
- degrees_id
  - INT
  - Not null
  - Foreign key (DEGREES id)
- degree_categories_id
  - INT
  - Not null
  - Primary key (degrees_id, degree_categories_id)
  - Foreign key (DEGREE_CATEGORIES id)
- credit
  - Decimal(4,2)
  - Not null

```sql
CREATE TABLE [dbo].[DEGREE_ADMISSION_CATEGORIES] (
[degrees_id] INT NOT NULL,
[degree_categories_id] INT NOT NULL,
[credit] DECIMAL (4, 2) NOT NULL,
CONSTRAINT [PK_DEGREE_ADMISSION_CATEGORIES] PRIMARY KEY CLUSTERED ([degrees_id]
ASC, [degree_categories_id] ASC),
CONSTRAINT [FK_DEGREE_ADMISSION_CATEGORIES_degrees_id] FOREIGN KEY ([degrees_id])
REFERENCES [dbo].[DEGREES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_DEGREE_ADMISSION_CATEGORIES_degree_categories_id] FOREIGN KEY
([degree_categories_id]) REFERENCES [dbo].[DEGREE_CATEGORIES] ([id])
);
```

DEGREE_GRADUATION_COURSES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm

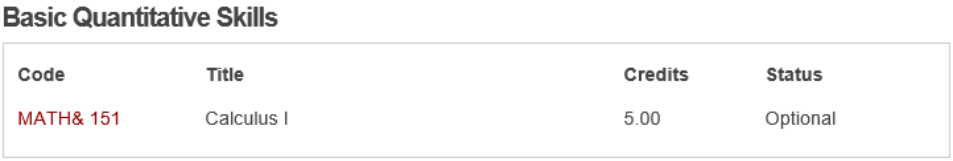


Optional Graduation Requirements

By Course

CC Course	Category	Equivalent Course	
ENGL& 101	Basic Communication Skills	English Composition I	Remove
MATH& 151	Basic Quantitative Skills	Calculus I	Remove

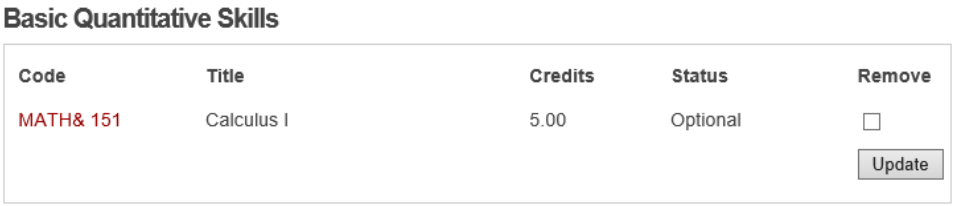
- AdvisorPortal\dashboard\index.cfm



Basic Quantitative Skills

Code	Title	Credits	Status
MATH& 151	Calculus I	5.00	Optional

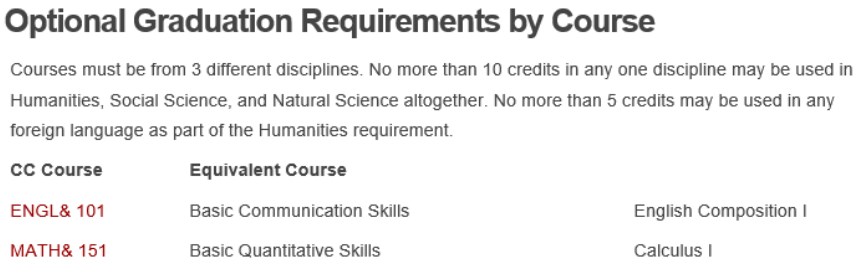
- AdvisorPortal\plans\edit\index.cfm



Basic Quantitative Skills

Code	Title	Credits	Status	Remove
MATH& 151	Calculus I	5.00	Optional	☐
				Update

- AdvisorPortal\view\degrees\index.cfm



Optional Graduation Requirements by Course

Courses must be from 3 different disciplines. No more than 10 credits in any one discipline may be used in Humanities, Social Science, and Natural Science altogether. No more than 5 credits may be used in any foreign language as part of the Humanities requirement.

CC Course	Equivalent Course	
ENGL& 101	Basic Communication Skills	English Composition I
MATH& 151	Basic Quantitative Skills	Calculus I

Used in the following stored procedures:

- createPLANS

Use to allow students to continue to map out EvCC classes beyond university degree admission requirements where specific EvCC courses can be optionally taken. These courses can be taken at the community college level instead of at the university. The courses_id column is the CC equivalency for the university specified course where the foreign_course_number is a label identifying the name of the university course for informational purposes only. Each college may only identify a courses_id once.

- degrees_id

    - INT
    - Not null
    - Foreign key (DEGREES id)
- courses_id
    - INT
    - Not null
    - Primary key (degrees_id, courses_id)
    - Foreign key (COURSES id)
- degree_categories_id
    - INT
    - Not null
    - Foreign key (DEGREE_CATEGORIES id)
- foreign_course_number
    - VARCHAR(50)
    - Not null

```sql
CREATE TABLE [dbo].[DEGREE_GRADUATION_COURSES] (
[degrees_id] INT NOT NULL,
[courses_id] INT NOT NULL,
[degree_categories_id] INT NOT NULL,
[foreign_course_number] VARCHAR (50) NOT NULL,
CONSTRAINT [PK_DEGREE_GRADUATION_COURSES] PRIMARY KEY CLUSTERED ([degrees_id] ASC,
[courses_id] ASC),
CONSTRAINT [FK_DEGREE_GRADUATION_COURSES_courses_id] FOREIGN KEY ([courses_id])
REFERENCES [dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_DEGREE_GRADUATION_COURSES_degree_categories_id] FOREIGN KEY
([degree_categories_id]) REFERENCES [dbo].[DEGREE_CATEGORIES] ([id]),
CONSTRAINT [FK_DEGREE_GRADUATION_COURSES_degrees_id] FOREIGN KEY ([degrees_id])
REFERENCES [dbo].[DEGREES] ([id])
);
```

DEGREE_GRADUATION_CATEGORIES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm

By Degree Category

CC category	Credit required	
Basic Communication Skills	10.00	Remove
Basic Quantitative Skills	5.00	Remove

- AdvisorPortal\view\degrees\index.cfm

Optional Graduation Requirements by Degree Category

No more than 5 credits in Humanities Performance may be used in Humanities. One lab science cla
used for Natural Sciences.

CC Codekey	Credit Required
Basic Communication Skills	10.00
Basic Quantitative Skills	5.00

// Use to allow students to continue taking EvCC classes beyond university admission requirements.
// Use when the graduation requirements of a degree call out types of courses that need to be taken.
// Example: college requires 15 credits of social sciences.

- degrees_id

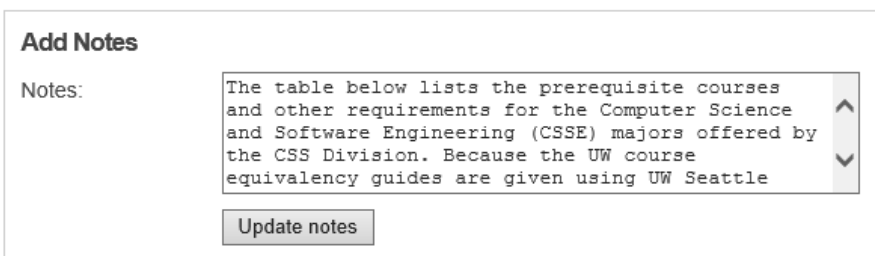
    - INT
    - Not null
    - Foreign key (DEGREES id)
- degree_categories_id
    - INT
    - Not null
    - Primary key (degrees_id, degree_categories_id)
    - Foreign key (DEGREE_CATEGORIES id)
- credit
    - Decimal(4,2)
    - Not null

```
CREATE TABLE [dbo].[DEGREE_GRADUATION_CATEGORIES] (
[degrees_id] INT NOT NULL,
[degree_categories_id] INT NOT NULL,
[credit] DECIMAL (4, 2) NOT NULL,
CONSTRAINT [PK_DEGREE_GRADUATION_CATEGORIES] PRIMARY KEY CLUSTERED ([degrees_id]
ASC, [degree_categories_id] ASC),
CONSTRAINT [FK_DEGREE_GRADUATION_CATEGORIES_degrees_id] FOREIGN KEY ([degrees_id])
REFERENCES [dbo].[DEGREES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_DEGREE_GRADUATION_CATEGORIES_degree_categories_id] FOREIGN KEY
([degree_categories_id]) REFERENCES [dbo].[DEGREE_CATEGORIES] ([id])
);
```

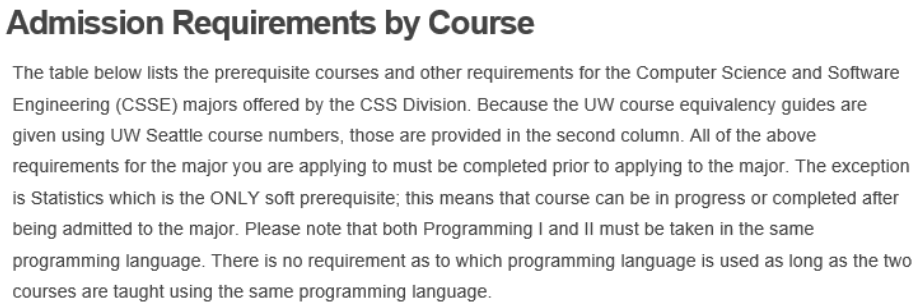
DEGREE_NOTES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm



- AdvisorPortal\view\degrees\index.cfm



Used in the following triggers:

- Trigger_COLLEGES_insert

Store notes relating to degrees that are viewable to advisors and students. For any degree, notes may exist for required admission or graduation courses or category (user defined categories per degree, such as VLPA, I&S, General Electives, etc).

- degrees_id
    - INT
    - Not null

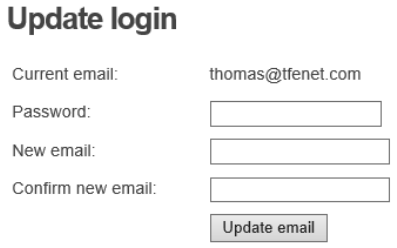
    - Primary key
    - Foreign key (DEGREES id)
- admission_courses_note
    - VARCHAR(1800)
- admission_categories_note
    - VARCHAR(1800)
- graduation_courses_note
    - VARCHAR(1800)
- graduation_categories_note
    - VARCHAR(1800)
- general_note
    - VARCHAR(1800)

```sql
CREATE TABLE [dbo].[DEGREE_NOTES] (
[degrees_id] INT NOT NULL,
[admission_courses_note] VARCHAR (1800) NULL,
[admission_categories_note] VARCHAR (1800) NULL,
[graduation_courses_note] VARCHAR (1800) NULL,
[graduation_categories_note] VARCHAR (1800) NULL,
[general_note] VARCHAR (1800) NULL,
CONSTRAINT [PK_DEGREE_NOTES] PRIMARY KEY CLUSTERED ([degrees_id] ASC),
CONSTRAINT [FK_DEGREE_NOTES_degrees_id] FOREIGN KEY ([degrees_id]) REFERENCES
[dbo].[DEGREES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

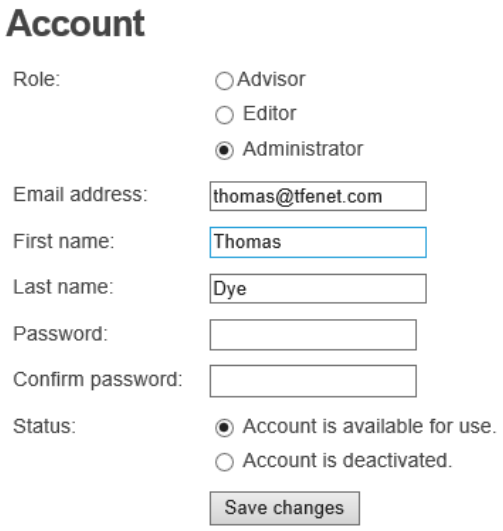
ACCOUNTS [Table]

Used in the following pages:

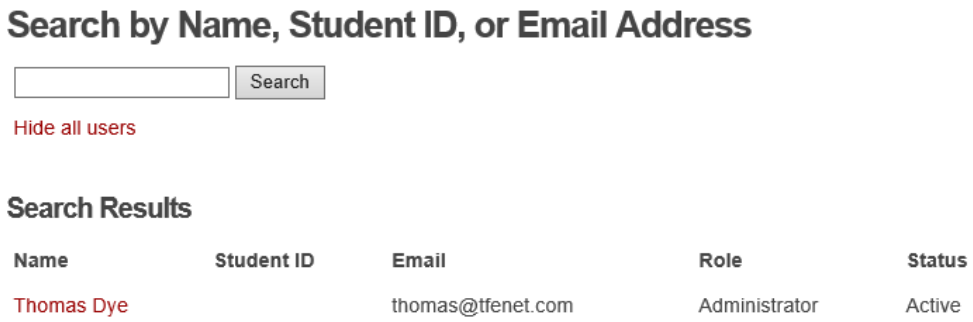
- AdvisorPortal\account\index.cfm



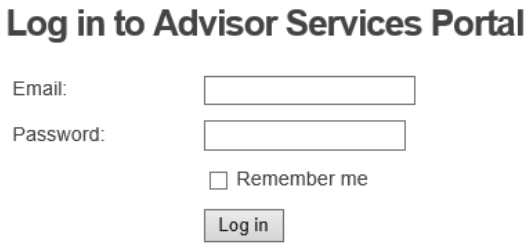
- AdvisorPortal\admin\manage-users\edit\index.cfm



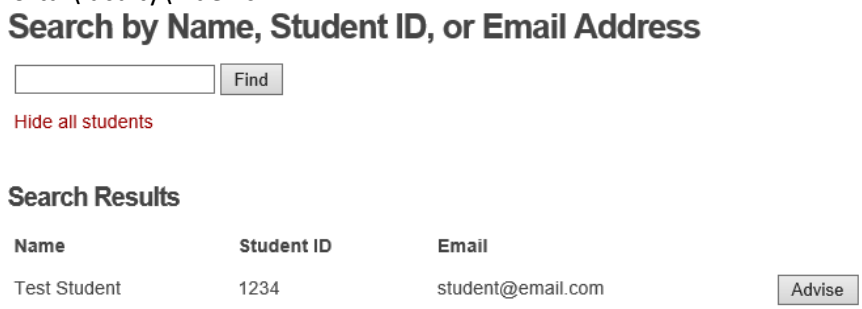
- AdvisorPortal\admin\manage-users\index.cfm



- AdvisorPortal\Application.cfc



- AdvisorPortal\faculty\index.cfm



Used in the following stored procedures:

- createPLANS
- updatePLAN_SELECTEDCOURSES

Contains the base account for all students, faculty, and staff. Passwords are stored as hashed values and are based off of the random salt. Accounts are disabled and prevented from logging in by unsetting the 'active' bit.

- id
  - INT
  - Auto increment
  - Primary key
- active
  - BIT
  - Not null
- email
  - VARCHAR(60)
  - Unique
  - Not null
- first_name
  - VARCHAR(40)
  - Not null
- last_name
  - VARCHAR(40)
  - Not null
- password
  - VARCHAR(255)
  - Not null

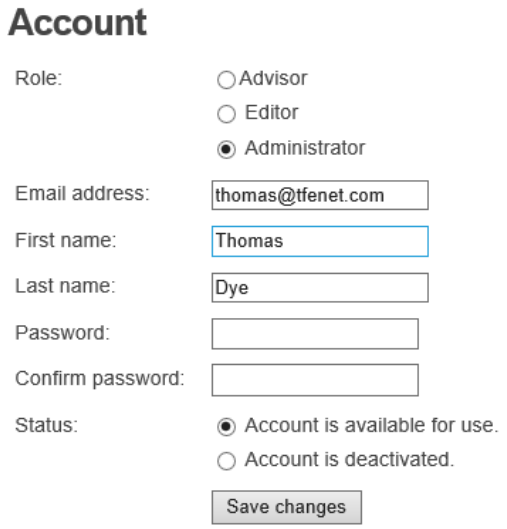
- salt
    - VARCHAR(255)
    - Not null

```
CREATE TABLE [dbo].[ACCOUNTS] (
[id] INT IDENTITY (1, 1) NOT NULL,
[active] BIT NOT NULL,
[email] VARCHAR (60) NOT NULL,
[first_name] VARCHAR (40) NOT NULL,
[last_name] VARCHAR (40) NOT NULL,
[password] VARCHAR (255) NOT NULL,
[salt] VARCHAR (255) NOT NULL,
CONSTRAINT [PK_ACCOUNTS] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [AK_ACCOUNTS_email] UNIQUE NONCLUSTERED ([email] ASC)
);
```

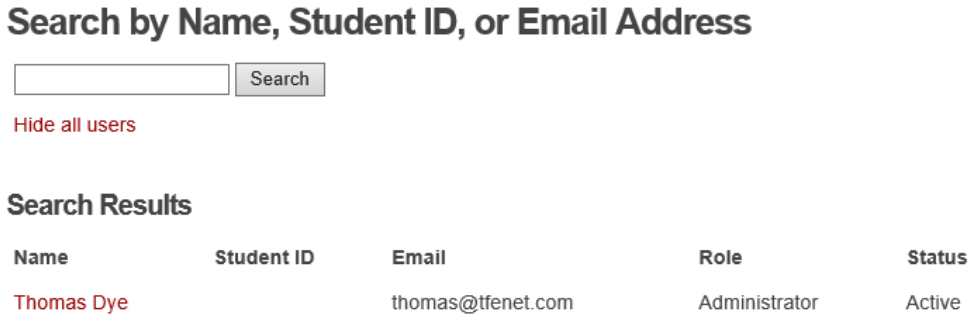
FACULTY [Table]

Used in the following pages:

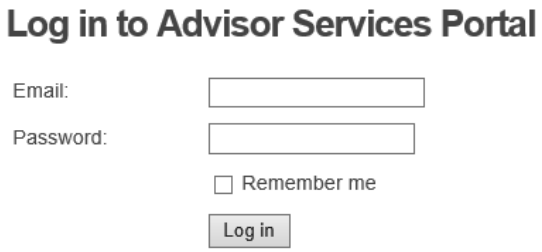
- AdvisorPortal\admin\manage-users\edit\index.cfm



- AdvisorPortal\admin\manage-users\index.cfm



- AdvisorPortal\Application.cfc



The faculty table extends account information and holds attributes that are unique to staff and faculty. If a record exists here, the account is an “advisor” by default. If the editor bit is set, the account is an “editor.” If the administrator bit is set, the account is an “administrator.” An account should either be faculty OR student, but not both. Website application logic will ignore this table if the account is marked as a student.

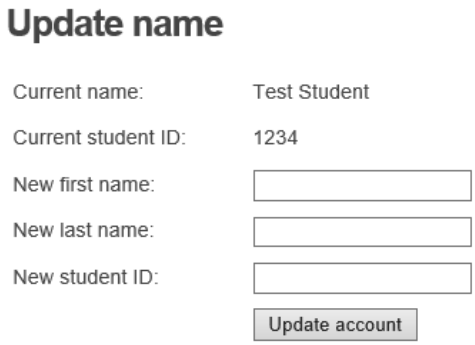
- accounts_id
  - INT
  - Not null
  - Primary key
  - Foreign key (ACCOUNTS id)
- editor
  - BIT
  - Not null
- administrator
  - BIT
  - Not null

```
CREATE TABLE [dbo].[FACULTY] (
[accounts_id] INT NOT NULL,
[editor] BIT NOT NULL,
[administrator] BIT NOT NULL,
CONSTRAINT [PK_FACULTY] PRIMARY KEY CLUSTERED ([accounts_id] ASC),
CONSTRAINT [FK_FACULTY_accounts_id] FOREIGN KEY ([accounts_id]) REFERENCES
[dbo].[ACCOUNTS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

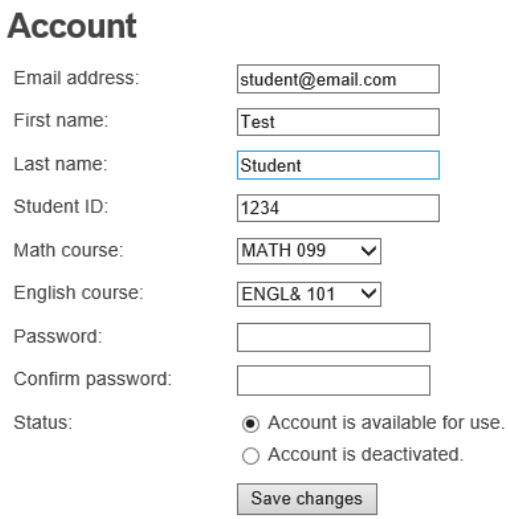
STUDENTS [Table]

Used in the following pages:

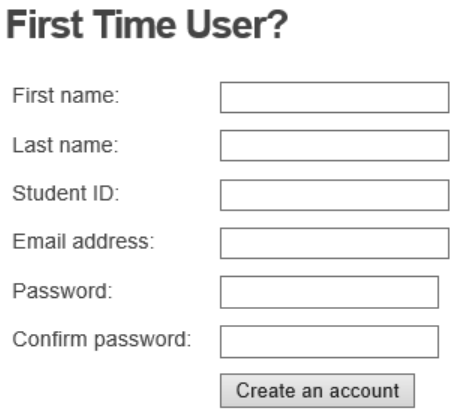
- AdvisorPortal\account\index.cfm



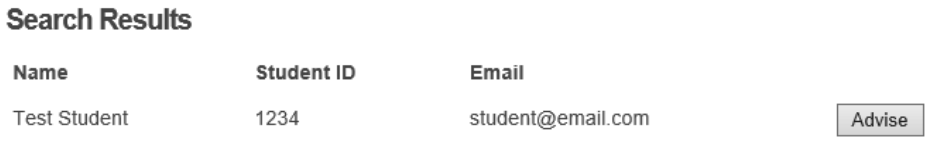
- AdvisorPortal\admin\manage-users\edit\index.cfm



- AdvisorPortal\Application.cfc



- AdvisorPortal\faculty\index.cfm



The students table extends account information and holds attributes that are unique to students. If a record exists here, the account is marked as a student. An account should either be student OR faculty, but not both. Website application logic will ignore faculty attributes if the account is marked as a student.

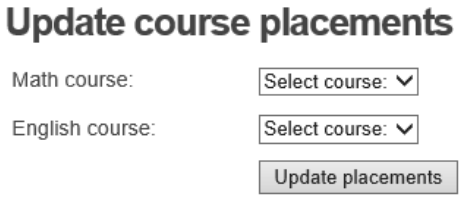
- accounts_id
  - INT
  - Not null
  - Primary key
  - Foreign key (ACCOUNTS id)
- student_id
  - INT
  - Not null
  - Unique

```
CREATE TABLE [dbo].[STUDENTS] (
[accounts_id] INT NOT NULL,
[student_id] INT NOT NULL,
CONSTRAINT [PK_STUDENTS] PRIMARY KEY CLUSTERED ([accounts_id] ASC),
CONSTRAINT [AK_STUDENTS_student_id] UNIQUE NONCLUSTERED ([student_id] ASC),
CONSTRAINT [FK_STUDENTS_accounts_id] FOREIGN KEY ([accounts_id]) REFERENCES
[dbo].[ACCOUNTS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

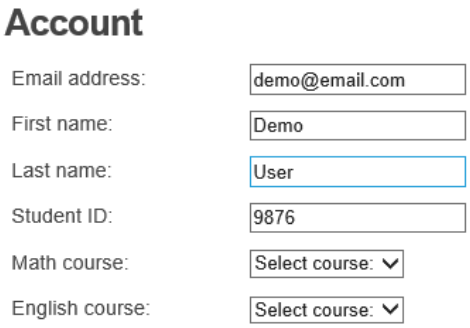
STUDENT_PLACEMENTCOURSES [Table]

Used in the following pages:

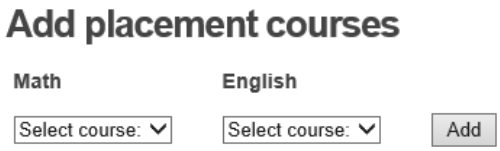
- AdvisorPortal\account\index.cfm



- AdvisorPortal\admin\manage-users\edit\index.cfm



- AdvisorPortal\dashboard\index.cfm



Track an English and Math placement course for each student, based off of Compass test results. Allows the recommendation engine to have a defined "stop" point when back-tracking courses and no applicable STUDENTS_COMPLETEDCOURSES records exist.

- students_accounts_id
  - INT
  - Not null
  - Primary key
  - Foreign key (STUDENTS accounts_id)
- math_courses_id
  - INT
  - Not null
  - Foreign key (COURSES id)
- english_courses_id
  - INT
  - Not null
  - Foreign key (COURSES id)

```sql
CREATE TABLE [dbo].[STUDENT_PLACEMENTCOURSES] (
[students_accounts_id] INT NOT NULL,
[math_courses_id] INT NOT NULL,
[english_courses_id] INT NOT NULL,
CONSTRAINT [PK_STUDENT_PLACEMENTCOURSES] PRIMARY KEY CLUSTERED
([students_accounts_id] ASC),
CONSTRAINT [FK_STUDENT_PLACEMENTCOURSES_math_courses_id] FOREIGN KEY
([math_courses_id]) REFERENCES [dbo].[COURSES] ([id]),
CONSTRAINT [FK_STUDENT_PLACEMENTCOURSES_english_courses_id] FOREIGN KEY
([english_courses_id]) REFERENCES [dbo].[COURSES] ([id]),
CONSTRAINT [FK_STUDENT_PLACEMENTCOURSES_students_accounts_id] FOREIGN KEY
([students_accounts_id]) REFERENCES [dbo].[STUDENTS] ([accounts_id]) ON DELETE
CASCADE ON UPDATE CASCADE
);
```

STUDENTS_COMPLETEDCOURSES [Table]

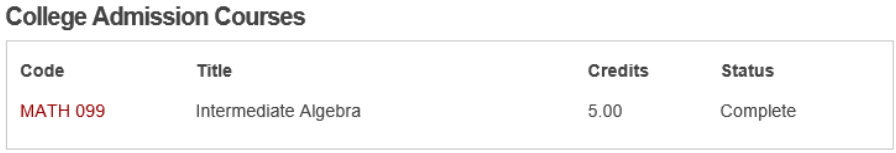
Used in the following pages:

- AdvisorPortal\courses\index.cfm

Completed Courses

Number	Title	Credits	
ENGL& 101	English Composition I	5.00	Delete
MATH 099	Intermediate Algebra	5.00	Delete

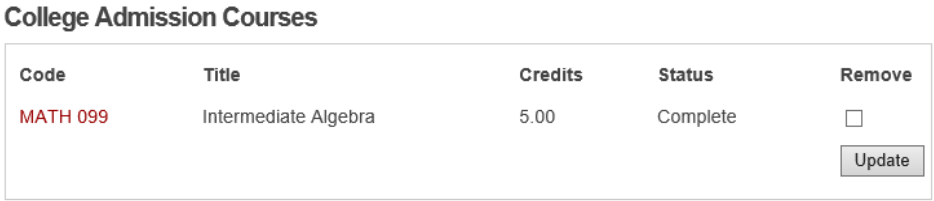
- AdvisorPortal\dashboard\index.cfm



College Admission Courses

Code	Title	Credits	Status
MATH 099	Intermediate Algebra	5.00	Complete

- AdvisorPortal\plans\edit\index.cfm



College Admission Courses

Code	Title	Credits	Status	Remove
MATH 099	Intermediate Algebra	5.00	Complete	☐
				Update

Used in the following stored procedures:

- updatePLAN_SELECTEDCOURSES

Track courses that a student marks as passed and completed. The recommendation engine should use this list to define "stop" points where back-tracking courses should finish. This table is intentionally constructed to allow courses to be repeated for additional credit, where applicable.

- id
  - INT
  - Auto increment
  - Not null
  - Primary key
- students_accounts_id
  - INT
  - Not null
  - Foreign key (STUDENTS accounts_id)
- courses_id
  - INT

    - Not null
    - Foreign key (COURSES id)
- credit
    - Decimal (4,2)
    - Not null

```
CREATE TABLE [dbo].[STUDENTS_COMPLETEDCOURSES] (
[id] INT IDENTITY (1, 1) NOT NULL,
[students_accounts_id] INT NOT NULL,
[courses_id] INT NOT NULL,
[credit] DECIMAL (4, 2) NOT NULL,
CONSTRAINT [PK_STUDENTS_COMPLETEDCOURSES] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_STUDENTS_COMPLETEDCOURSES_courses_id] FOREIGN KEY ([courses_id])
REFERENCES [dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_STUDENTS_COMPLETEDCOURSES_students_accounts_id] FOREIGN KEY
([students_accounts_id]) REFERENCES [dbo].[STUDENTS] ([accounts_id]) ON DELETE
CASCADE ON UPDATE CASCADE
);

GO
-- Thomas Dye, September 2016 --
CREATE TRIGGER [dbo].[Trigger_STUDENTS_COMPLETEDCOURSES_insert]
ON [dbo].[STUDENTS_COMPLETEDCOURSES]
FOR INSERT
AS
	BEGIN
		SET NoCount ON

		--Do book keeping for COURSE_RANKINGS --
		DECLARE @CourseID INT
		SET @CourseID = (SELECT c.id
			FROM COURSES c
			JOIN inserted i ON c.id = i.courses_id)
		IF EXISTS (SELECT courses_id
			FROM COURSE_RANKINGS
			WHERE courses_id = @CourseID)
		BEGIN
			DECLARE @CourseRank AS INT
			SET @CourseRank = (SELECT rank
				FROM COURSE_RANKINGS
				WHERE courses_id = @CourseID)
			SET @CourseRank = @CourseRank + 1
			UPDATE COURSE_RANKINGS
				SET rank = @CourseRank
				WHERE courses_id = @CourseID
		END
		IF NOT EXISTS (SELECT courses_id
			FROM COURSE_RANKINGS
			WHERE courses_id = @CourseID)
		BEGIN
			INSERT
				INTO COURSE_RANKINGS (courses_id, rank)
				VALUES (@CourseID, 1)
		END
		-- Update completedcourse_id in PLAN_SELECTEDCOURSES for this
student's plans --
		DECLARE @AccountID INT
		SET @AccountID = (SELECT students_accounts_id FROM inserted)
```

```
            -- Create an iterator to update all the student's plans for this
completed course --
            DECLARE @PlanCursor CURSOR
            SET @PlanCursor = CURSOR READ_ONLY FOR
                  SELECT id FROM PLANS WHERE students_accounts_id = @AccountID
            OPEN @PlanCursor
            -- Variable to store content of current plan being evaluated --
            DECLARE @PlanID INT
            -- Loop through the student's plans until there are none left --
            FETCH NEXT FROM @PlanCursor INTO @PlanID
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  EXEC updatePLAN_SELECTEDCOURSES @AccountID, @PlanID
                  FETCH NEXT FROM @PlanCursor INTO @PlanID
            END
            CLOSE @PlanCursor
            DEALLOCATE @PlanCursor
      END
GO
-- Thomas Dye, September 2016 --
CREATE TRIGGER [dbo].[Trigger_STUDENTS_COMPLETEDCOURSES_delete]
ON [dbo].[STUDENTS_COMPLETEDCOURSES]
FOR DELETE
AS
BEGIN
SET NoCount ON
            --Do book keeping for COURSE_RANKINGS --
            DECLARE @CourseID INT
            SET @CourseID = (SELECT c.id
                  FROM COURSES c
                  JOIN deleted d ON c.id = d.courses_id)
            IF EXISTS (SELECT courses_id
                  FROM COURSE_RANKINGS
                  WHERE courses_id = @CourseID)
            BEGIN
                  DECLARE @CourseRank AS INT
                  SET @CourseRank = (SELECT rank
                        FROM COURSE_RANKINGS
                        WHERE courses_id = @CourseID)
                  IF (@CourseRank <> 0)
                  BEGIN
                        SET @CourseRank = @CourseRank - 1
                        UPDATE COURSE_RANKINGS
                              SET rank = @CourseRank
                              WHERE courses_id = @CourseID
                  END
            END
            -- Do book keeping for PLAN_SELECTEDCOURSES --
            DECLARE @AccountID INT
            SET @AccountID = (SELECT students_accounts_id FROM deleted)
            -- Create an iterator to update all the student's plans for this
completed course --
            DECLARE @PlanCursor CURSOR
            SET @PlanCursor = CURSOR READ_ONLY FOR
                  SELECT id FROM PLANS WHERE students_accounts_id = @AccountID
            OPEN @PlanCursor
            -- Variable to store content of current plan being evaluated --
            DECLARE @PlanID INT
            -- Loop through the student's plans until there are none left --
```

```
            FETCH NEXT FROM @PlanCursor INTO @PlanID
            WHILE @@FETCH_STATUS = 0
            BEGIN
                  UPDATE PLAN_SELECTEDCOURSES SET completedcourses_id = NULL
WHERE completedcourses_id IN (SELECT id FROM deleted) AND plans_id = @PlanID
                  FETCH NEXT FROM @PlanCursor INTO @PlanID
            END
            CLOSE @PlanCursor
            DEALLOCATE @PlanCursor
END
```

PLANS [Table]

Used in the following pages:

- AdvisorPortal\dashboard\index.cfm

  Active Plan: Computer Science and Software Engineering

  Computer Science and Software Engineering
  University of Washington - Bothell
  Bachelor of Science

- AdvisorPortal\plans\edit\index.cfm

  Basic Details

  Degree: Computer Science and Software Engineering
  University of Washington - Bothell
  Bachelor of Science

- AdvisorPortal\plans\index.cfm

  Active Degree Plan

  Computer Science and Software Engineering
  University of Washington - Bothell
  Bachelor of Science

Used in the following triggers:

- Trigger_STUDENTS_COMPLETEDCOURSES_insert
- Trigger_STUDENTS_COMPLETEDCOURSES_delete
- Trigger_PLAN_SELECTEDDEGREES_insert_update
- Trigger_PLAN_SELECTEDCOURSES_insert_update
- Trigger_PLAN_SELECTEDCOURSES_delete

Used in the following stored procedures:

- createPLANS
- updatePLAN_SELECTEDCOURSES

Associates a degree plan with a student. Students may have multiple degree plans to explore various education options. Example: The intention to transfer to UWB's CSSE program is a degree plan. The courses and scheduling of courses as part of that degree plan are not contained in this table. Degree plans are marked as updated through triggers for listing purposes.

- id
  - INT
  - Auto increment
  - Not null
  - Primary key
- students_accounts_id
  - INT
  - Not null

    - Foreign key (STUDENTS accounts_id)
- plan_name
    - VARCHAR(255)
    - Not null
- time_created
    - DATETIME
    - Default current time
    - Not null
- time_updated
    - DATETIME

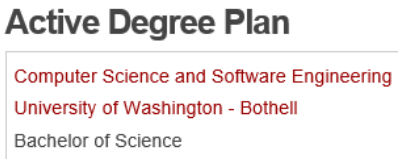
```
CREATE TABLE [dbo].[PLANS] (
[id] INT IDENTITY (1, 1) NOT NULL,
[students_accounts_id] INT NOT NULL,
[plan_name] VARCHAR (255) NOT NULL,
[time_created] DATETIME DEFAULT (getdate()) NOT NULL,
[time_updated] DATETIME NULL,
CONSTRAINT [PK_PLANS] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_PLANS_students_accounts_id] FOREIGN KEY ([students_accounts_id])
REFERENCES [dbo].[STUDENTS] ([accounts_id]) ON DELETE CASCADE ON UPDATE CASCADE
);
```

PLAN_ACTIVEPLANS [Table]

Used in the following pages:

- AdvisorPortal\dashboard\index.cfm

- AdvisorPortal\plans\index.cfm



Used in the following stored procedures:

- createPLANS

Defines the active (plan in use) by a student, which shows up on the student dashboard. A student may not have more than one active plan. The first degree plan a student creates will become the active plan. Additional plans can be changed into the active plan through UI selection. The active plan cannot be deleted, rather, it can only be changed. Changing the active plan affects department, college, and degree rankings within the system, potentially changing their popularity sort orders for when new degree plans are listed.

- plans_id
    - INT
    - Not null
    - Foreign key (PLANS id)
- students_acccounts_id
    - INT
    - Not null
    - Primary key (plans_id, students_accounts_id)
    - Foreign key (STUDENTS accounts_id)

```sql
CREATE TABLE [dbo].[PLAN_ACTIVEPLANS] (
[plans_id] INT NOT NULL,
[students_accounts_id] INT NOT NULL,
CONSTRAINT [PK_PLAN_ACTIVEPLANS] PRIMARY KEY CLUSTERED ([plans_id] ASC,
[students_accounts_id] ASC),
CONSTRAINT [FK_PLAN_ACTIVEPLANS_students_accounts_id] FOREIGN KEY
([students_accounts_id]) REFERENCES [dbo].[STUDENTS] ([accounts_id]),
CONSTRAINT [FK_PLAN_ACTIVEPLANS_plans_id] FOREIGN KEY ([plans_id]) REFERENCES
[dbo].[PLANS] ([id])
);

GO
-- Thomas Dye, September 2016 --
CREATE TRIGGER [dbo].[Trigger_PLAN_ACTIVEPLANS_insert]
ON [dbo].[PLAN_ACTIVEPLANS]
FOR INSERT
AS
BEGIN
SET NoCount ON
             BEGIN
                    DECLARE @PlansID INT
                    SET @PlansID = (SELECT
                           inserted.plans_id
                           FROM inserted)
                    --Get degree, college, and department associated with plan
                    DECLARE @DegreeID INT
                    DECLARE @CollegeID INT
                    DECLARE @DepartmentID INT
                    SET @DegreeID = (SELECT degrees_id
                           FROM PLAN_SELECTEDDEGREES
                           WHERE plans_id = @PlansID)
                    SET @CollegeID = (SELECT colleges_id
                           FROM DEGREES
                           WHERE id = @DegreeID)
                    SET @DepartmentID = (SELECT departments_id
                           FROM DEGREES
                           WHERE id = @DegreeID)

                    --Update degree rankings
                    IF EXISTS (SELECT degrees_id
                           FROM DEGREE_RANKINGS
                           WHERE degrees_id = @DegreeID)
                    BEGIN
                           DECLARE @DegreeRank INT
                           SET @DegreeRank = (SELECT rank
                                  FROM DEGREE_RANKINGS
                                  WHERE degrees_id = @DegreeID)
                           SET @DegreeRank = @DegreeRank + 1
                           UPDATE DEGREE_RANKINGS
                                  SET rank = @DegreeRank
                                  WHERE degrees_id = @DegreeID
                    END
                    ELSE
                    BEGIN
                           INSERT
                                  INTO DEGREE_RANKINGS (degrees_id, rank)
                                  VALUES (@DegreeID, 1)
                    END
                    --Update college rankings
```

```
                IF EXISTS (SELECT colleges_id
                        FROM COLLEGE_RANKINGS
                        WHERE colleges_id = @CollegeID)
                BEGIN
                        DECLARE @CollegeRank INT
                        SET @CollegeRank = (SELECT rank
                                FROM COLLEGE_RANKINGS
                                WHERE colleges_id = @CollegeID)
                        SET @CollegeRank = @CollegeRank + 1
                        UPDATE COLLEGE_RANKINGS
                                SET rank = @CollegeRank
                                WHERE colleges_id = @CollegeID
                END
                ELSE
                BEGIN
                        INSERT
                                INTO COLLEGE_RANKINGS (colleges_id, rank)
                                VALUES (@CollegeID, 1)
                END
                --Update department rankings
                IF EXISTS (SELECT departments_id
                        FROM DEPARTMENT_RANKINGS
                        WHERE departments_id = @DepartmentID)
                BEGIN
                        DECLARE @DepartmentRank INT
                        SET @DepartmentRank = (SELECT rank
                                FROM DEPARTMENT_RANKINGS
                                WHERE departments_id = @DepartmentID)
                        SET @DepartmentRank = @DepartmentRank + 1
                        UPDATE DEPARTMENT_RANKINGS
                                SET rank = @DepartmentRank
                                WHERE departments_id = @DepartmentID
                END
                ELSE
                BEGIN
                        INSERT
                                INTO DEPARTMENT_RANKINGS (departments_id, rank)
                                VALUES (@DepartmentID, 1)
                END
        END
END
GO
-- Thomas Dye, September 2016 --
CREATE TRIGGER [dbo].[Trigger_PLAN_ACTIVEPLANS_delete]
ON [dbo].[PLAN_ACTIVEPLANS]
FOR DELETE
AS
BEGIN
SET NoCount ON
        BEGIN
                DECLARE @PlansID INT
                SET @PlansID = (SELECT deleted.plans_id
                        FROM deleted)
                --Get degree, college, and department associated with plan
                DECLARE @DegreeID INT
                DECLARE @CollegeID INT
                DECLARE @DepartmentID INT
                SET @DegreeID = (SELECT degrees_id
                        FROM PLAN_SELECTEDDEGREES
```

```
                WHERE plans_id = @PlansID)
        SET @CollegeID = (SELECT colleges_id
                FROM DEGREES
                WHERE id = @DegreeID)
        SET @DepartmentID = (SELECT departments_id
                FROM DEGREES
                WHERE id = @DegreeID)
        --Update degree rankings
        IF EXISTS (SELECT degrees_id
                FROM DEGREE_RANKINGS
                WHERE degrees_id = @DegreeID)
        BEGIN
                DECLARE @DegreeRank INT
                SET @DegreeRank = (SELECT rank
                        FROM DEGREE_RANKINGS
                        WHERE degrees_id = @DegreeID)
                IF (@DegreeRank <> 0)
                BEGIN
                        SET @DegreeRank = @DegreeRank - 1
                        UPDATE DEGREE_RANKINGS
                                SET rank = @DegreeRank
                                WHERE degrees_id = @DegreeID
                END
        END
        --Update college rankings
        IF EXISTS (SELECT colleges_id
                FROM COLLEGE_RANKINGS
                WHERE colleges_id = @CollegeID)
        BEGIN
                DECLARE @CollegeRank INT
                SET @CollegeRank = (SELECT rank
                        FROM COLLEGE_RANKINGS
                        WHERE colleges_id = @CollegeID)
                IF (@CollegeRank <> 0)
                BEGIN
                        SET @CollegeRank = @CollegeRank - 1
                        UPDATE COLLEGE_RANKINGS
                                SET rank = @CollegeRank
                                WHERE colleges_id = @CollegeID
                END
        END
        --Update department rankings
        IF EXISTS (SELECT departments_id
                FROM DEPARTMENT_RANKINGS
                WHERE departments_id = @DepartmentID)
        BEGIN
                DECLARE @DepartmentRank INT
                SET @DepartmentRank = (SELECT rank
                        FROM DEPARTMENT_RANKINGS
                        WHERE departments_id = @DepartmentID)
                IF (@DepartmentRank <> 0)
                BEGIN
                        SET @DepartmentRank = @DepartmentRank - 1
                        UPDATE DEPARTMENT_RANKINGS
                                SET rank = @DepartmentRank
                                WHERE departments_id = @DepartmentID
                END
        END
END
```

```
END
```

PLAN_SELECTEDDEGREES [Table]

Used in the following pages:

- AdvisorPortal\dashboard\index.cfm

  Active Plan: Computer Science and Software Engineering

  Computer Science and Software Engineering
  University of Washington - Bothell
  Bachelor of Science

- AdvisorPortal\plans\edit\index.cfm

  Basic Details

  Degree: Computer Science and Software Engineering
  University of Washington - Bothell
  Bachelor of Science

- AdvisorPortal\plans\index.cfm

  Saved Degree Plans

Plan details	Created	Updated		
Computer Science and Software Engineering	9:08 PM - 3/5/17	7:53 PM - 3/13/17	Delete	Copy
Computer Science Transfer - UWB	6:55 PM - 3/10/17	7:53 PM - 3/13/17	Delete	Copy

Used in the following triggers:

- Trigger_PLAN_ACTIVEPLANS_insert
- Trigger_PLAN_ACTIVEPLANS_delete

Used in the following stored procedures:

- createPLANS

Associates a transfer degree to a plan. There is only ever one degree tied to a plan. Triggers an update to PLANS time_updated column.

- plans_id
  - INT
  - Not null
  - Foreign key (PLANS id)
- degrees_id
  - INT
  - Not null
  - Primary key (plans_id, degrees_id)
  - Foreign key (DEGREES id)

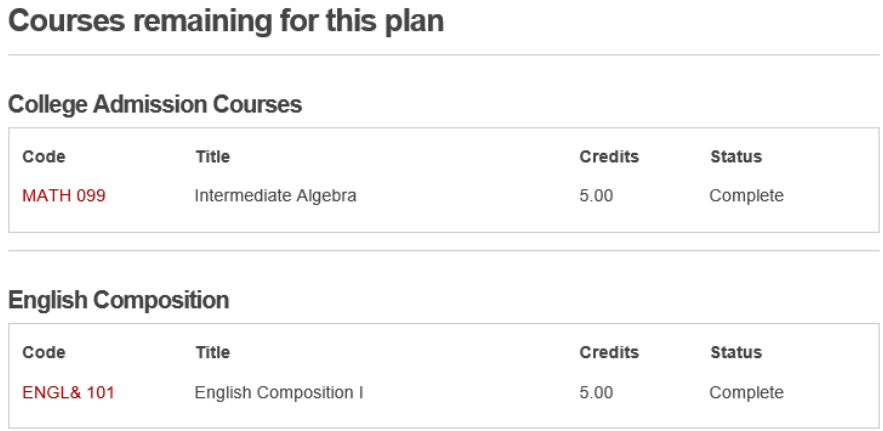
```
CREATE TABLE [dbo].[PLAN_SELECTEDDEGREES] (
[plans_id] INT NOT NULL,
[degrees_id] INT NOT NULL,
CONSTRAINT [PK_PLAN_SELECTEDDEGREES] PRIMARY KEY CLUSTERED ([plans_id] ASC,
[degrees_id] ASC),
CONSTRAINT [FK_PLAN_SELECTEDDEGREES_degrees_id] FOREIGN KEY ([degrees_id])
REFERENCES [dbo].[DEGREES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_PLAN_SELECTEDDEGREES_plans_id] FOREIGN KEY ([plans_id]) REFERENCES
[dbo].[PLANS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE
);

GO
-- Thomas Dye, February 2017 --
CREATE TRIGGER [dbo].[Trigger_PLAN_SELECTEDDEGREES_insert_update]
ON [dbo].[PLAN_SELECTEDDEGREES]
```

```
FOR INSERT, UPDATE
AS
BEGIN
SET NoCount ON
             UPDATE PLANS
                    SET time_updated = GETDATE()
                    WHERE id IN (SELECT inserted.plans_id
                           FROM inserted)
END
```

PLAN_SELECTEDCOURSES [Table]

Used in the following pages:

- AdvisorPortal\admin\manage-colleges\degrees\edit\index.cfm
- AdvisorPortal\dashboard\index.cfm



Courses remaining for this plan

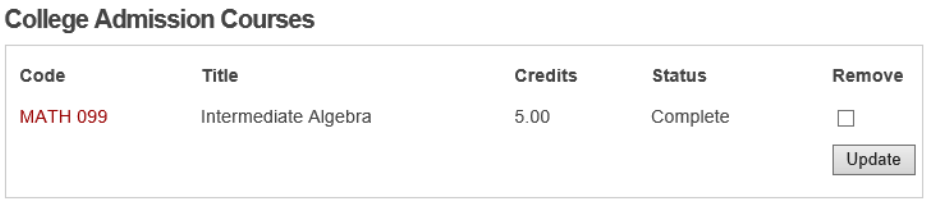
College Admission Courses

Code	Title	Credits	Status
MATH 099	Intermediate Algebra	5.00	Complete

English Composition

Code	Title	Credits	Status
ENGL& 101	English Composition I	5.00	Complete

- AdvisorPortal\plans\edit\index.cfm



College Admission Courses

Code	Title	Credits	Status	Remove
MATH 099	Intermediate Algebra	5.00	Complete	☐

Update

Used in the following triggers:

- Trigger_STUDENTS_COMPLETEDCOURSES_delete

Used in the following stored procedures:

- createPLANS
- updatePLAN_SELECTEDCOURSES

Location to aggregate all courses with a degree plan as all college admission, degree admission, and degree graduation requirements end up here. This should be the list evaluated and modified as appropriate by the recommendation engine when courses are added through backtracking. The completedcourses_id column is updated by stored procedure as STUDENTS_COMPLETEDCOURSES are reconciled against this table where the value is the primary key of STUDENTS_COMPLETEDCOURSES. The credit column is blank if courses are variable credit. A student must declare the amount of credit taken before STUDENTS_COMPLETEDCOURSES can be appropriately mapped. Updating this table triggers an update to PLANS time_updated.

- id
  - INT
  - Auto increment
  - Not null
  - Primary key

- plans_id
  - INT
  - Not null
  - Foreign key (PLANS id)
- courses_id
  - INT
  - Not null
  - Foreign key (COURSES id)
- degree_categories_id
  - INT
  - Not null
  - Foreign key (DEGREE_CATEGORIES id)
- completedcourses_id
  - INT
- credit
  - DECIMAL(4,2)

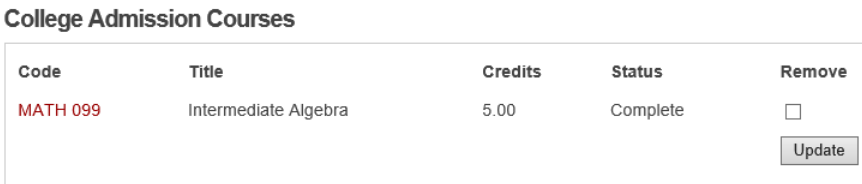
```
CREATE TABLE [dbo].[PLAN_SELECTEDCOURSES] (
[id] INT IDENTITY (1, 1) NOT NULL,
[plans_id] INT NOT NULL,
[courses_id] INT NOT NULL,
[degree_categories_id] INT NOT NULL,
[completedcourses_id] INT NULL,
[credit] DECIMAL (4, 2) NULL
CONSTRAINT [PK_PLAN_SELECTEDCOURSES] PRIMARY KEY CLUSTERED ([id] ASC),
CONSTRAINT [FK_PLAN_SELECTEDCOURSES_plans_id] FOREIGN KEY ([plans_id]) REFERENCES
[dbo].[PLANS] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_PLAN_SELECTEDCOURSES_courses_id] FOREIGN KEY ([courses_id])
REFERENCES [dbo].[COURSES] ([id]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT [FK_PLAN_SELECTEDCOURSES_degree_categories_id] FOREIGN KEY
([degree_categories_id]) REFERENCES [dbo].[DEGREE_CATEGORIES] ([id])
);

GO
-- Thomas Dye, September 2016 --
CREATE TRIGGER [dbo].[Trigger_PLAN_SELECTEDCOURSES_insert_update]
ON [dbo].[PLAN_SELECTEDCOURSES]
FOR INSERT, UPDATE
AS
BEGIN
SET NoCount ON
             UPDATE PLANS
                    SET time_updated = getdate()
                    WHERE id IN (SELECT inserted.plans_id
                           FROM inserted)
END
GO
-- Thomas Dye, September 2016 --
CREATE TRIGGER [dbo].[Trigger_PLAN_SELECTEDCOURSES_delete]
ON [dbo].[PLAN_SELECTEDCOURSES]
FOR DELETE
AS
BEGIN
SET NoCount ON
             UPDATE PLANS
                    SET time_updated = getdate()
                    WHERE id IN (SELECT deleted.plans_id
```

```
                                FROM deleted)
END
```

createPLANS [Stored Procedure]

Used in the following pages:

- AdvisorPortal\plans\create-plan\index.cfm



Aggregate all courses that are associated with a college and a degree when transferring to a university and add them to a student's plan. Any courses for a degree a gathered and sorted into their respective categories for a plan. Results end up in PLAN_SELECTEDCOURSES as a flat list and can be used to feed the backtracking algorithm for the recommendation engine.

```
-- Thomas Dye, February 2017 --
CREATE PROCEDURE [dbo].[createPLANS]
	@AccountID INT, -- from session
	@DegreeID INT, -- from form
	@PlanName VARCHAR(255) -- from form
AS
	SET NOCOUNT ON

	-- Validate all external input before proceeding --
	DECLARE @ValidAccount BIT = 0
	DECLARE @ValidDegree BIT = 0
	IF ((SELECT COUNT(*) FROM ACCOUNTS WHERE id = @AccountID) > 0)
		SET @ValidAccount = 1
	IF ((SELECT COUNT(*) FROM DEGREES WHERE id = @DegreeID) > 0)
		SET @ValidDegree = 1
	IF (@ValidAccount = 1 AND @ValidDegree = 1)
	BEGIN
		-- Create new student plan --
		INSERT INTO PLANS (
			students_accounts_id, plan_name
		) VALUES (
			@AccountID, @PlanName
		)
		-- Retrieve the new plan ID --
		DECLARE @PlanID INT = (SELECT DISTINCT @@IDENTITY AS id FROM PLANS)
		-- Associate the degree with the new plan --
		INSERT INTO PLAN_SELECTEDDEGREES (
			plans_id, degrees_id
		) VALUES (
			@PlanID, @DegreeID
		)
		-- If the student has no active plans, make this the active plan --
		IF ((SELECT COUNT(*) FROM PLAN_ACTIVEPLANS WHERE students_accounts_id
= @AccountID) = 0)
		BEGIN
			INSERT INTO PLAN_ACTIVEPLANS (
				plans_id, students_accounts_id
			) VALUES (
```

```
                        @PlanID, @AccountID
                )
        END
        -- Get all available admission and graduation courses for this degree
--
        -- Add the degree inherited courses to the plan --
        INSERT INTO PLAN_SELECTEDCOURSES (
                plans_id, courses_id, degree_categories_id, credit
        )
                SELECT @PlanID, cou.id, deg.degree_categories_id,
                        CASE
                                WHEN cou.min_credit IS NULL THEN cou.max_credit
                                ELSE NULL
                        END AS credit
                FROM COURSES cou
                JOIN DEGREE_ADMISSION_COURSES deg
                ON cou.id = deg.courses_id
                WHERE cou.use_catalog = 1
                AND deg.degrees_id = @DegreeID
                UNION
                SELECT @PlanID, cou.id, deg.degree_categories_id,
                        CASE
                                WHEN cou.min_credit IS NULL THEN cou.max_credit
                                ELSE NULL
                        END AS credit
                FROM COURSES cou
                JOIN DEGREE_GRADUATION_COURSES deg
                ON cou.id = deg.courses_id
                WHERE cou.use_catalog = 1
                AND deg.degrees_id = @DegreeID
        END
RETURN @PlanID
```

updatePLAN_SELECTEDCOURSES [Stored Procedure]

Used in the following pages:

- AdvisorPortal\plans\edit\index.cfm

College Admission Courses

Code	Title	Credits	Status
MATH 099	Intermediate Algebra	5.00	Complete

Used in the following triggers:

- Trigger_STUDENTS_COMPLETEDCOURSES_insert

This stored procedure will mark student courses as "Complete" in the web UI. It updates the PLAN_SELECTEDCOURSES table completedcourses_id column to reflect the value of a primary key value found in STUDENTS_COMPLETEDCOURSES if the key exists and also matches the credit declared in PLAN_SELECTEDCOURSES.

```
-- Update the completedcourses_id column of PLAN_SELECTEDCOURSES from a student's
completed courses --
-- Thomas Dye, February 2017 --
CREATE PROCEDURE [dbo].[updatePLAN_SELECTEDCOURSES]
        @AccountID int,
        @PlanID int
```

```
AS
	SET NOCOUNT ON

	-- Validate all external input before proceeding --
	DECLARE @ValidAccount BIT = 0
	DECLARE @ValidPlan BIT = 0
	IF ((SELECT COUNT(*) FROM ACCOUNTS WHERE id = @AccountID) > 0)
		SET @ValidAccount = 1
	IF ((SELECT COUNT(*) FROM PLANS WHERE id = @PlanID AND students_accounts_id
= @AccountID) > 0)
		SET @ValidPlan = 1
	IF (@ValidAccount = 1 AND @ValidPlan = 1)
	BEGIN
		-- Clear any previously marked completed courses in the student's
plan --
		UPDATE PLAN_SELECTEDCOURSES SET completedcourses_id = NULL WHERE
plans_id = @PlanID

		-- Create an iterator for the student's completed courses --
		DECLARE @CompletedCursor CURSOR
		SET @CompletedCursor = CURSOR READ_ONLY FOR
			SELECT id, courses_id, credit FROM STUDENTS_COMPLETEDCOURSES
WHERE students_accounts_id = @AccountID
		OPEN @CompletedCursor
		-- Variables to store contents of the current
STUDENTS_COMPLETEDCOURSES row --
		DECLARE @CompletedID INT
		DECLARE @CompletedCoursesID INT
		DECLARE @CompletedCredit DECIMAL(4,2)
		-- Loop through the completed courses until there are none left --
		FETCH NEXT FROM @CompletedCursor INTO @CompletedID,
@CompletedCoursesID, @CompletedCredit
		WHILE @@FETCH_STATUS = 0
		BEGIN
			-- Create a nested iterator for the student's selected courses
in a plan --
			-- Evaluate each planned course to see if it should be marked
as completed or not --
			DECLARE @SelectedCursor CURSOR
			SET @SelectedCursor = CURSOR FOR
				SELECT id, courses_id, completedcourses_id, credit FROM
PLAN_SELECTEDCOURSES WHERE plans_id = @PlanID
			OPEN @SelectedCursor
			-- Variables to store contents of PLANS_SELECTEDCOURSES row --
			DECLARE @SelectedID INT
			DECLARE @SelectedCoursesID INT
			DECLARE @SelectedCompletedCoursesID INT
			DECLARE @SelectedCredit INT
			-- Loop through the selected courses until there are none left
--
			FETCH NEXT FROM @SelectedCursor INTO @SelectedID,
@SelectedCoursesID, @SelectedCompletedCoursesID, @SelectedCredit
			WHILE @@FETCH_STATUS = 0
			BEGIN
				-- Match by course number and when planned credit equals
taken credit --
				IF (@CompletedCoursesID = @SelectedCoursesID AND
@CompletedCredit = @SelectedCredit AND @SelectedCompletedCoursesID IS NULL)
				BEGIN
```

```
                                UPDATE PLAN_SELECTEDCOURSES SET
completedcourses_id = @CompletedID WHERE id = @SelectedID
                          END
                          FETCH NEXT FROM @SelectedCursor INTO @SelectedID,
@SelectedCoursesID, @SelectedCompletedCoursesID, @SelectedCredit
                    END
                    CLOSE @SelectedCursor
                    DEALLOCATE @SelectedCursor
                    FETCH NEXT FROM @CompletedCursor INTO @CompletedID,
@CompletedCoursesID, @CompletedCredit
             END
             CLOSE @CompletedCursor
             DEALLOCATE @CompletedCursor
             RETURN 0
       END
RETURN -1
```