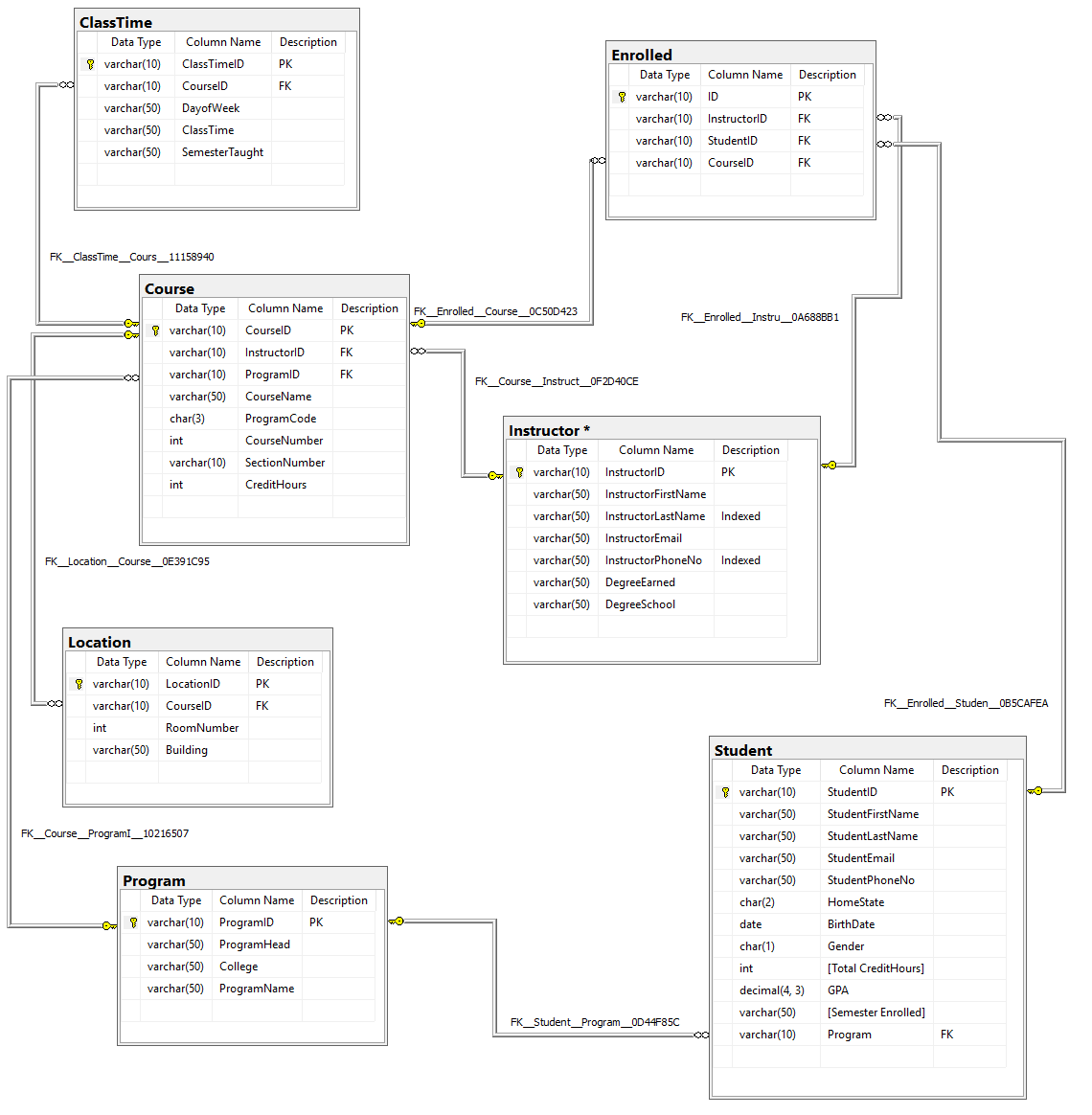
CIS 250 Final Project

Professor Ford

ERD



Database Creation

--Student Table

USE [am0742552]

GO

CREATE TABLE [dbo].[Student](

[StudentID] [varchar](10) NOT NULL,

[StudentFirstName] [varchar](50) NULL,

[StudentLastName] [varchar](50) NULL,

[StudentEmail] [varchar](50) NOT NULL,

[StudentPhoneNo] [varchar](50) NULL,

[HomeState] [char](2) NULL,

[BirthDate] [date] NOT NULL DEFAULT '1901-01-01',

[Gender] [char](1) NOT NULL,

[Total CreditHours] [int] NULL,

[GPA] [decimal](4, 3) NULL,

[Semester Enrolled] [varchar](50) NULL,

[Program] [varchar](10) NULL,

CONSTRAINT CK\_PhoneNumber CHECK (StudentPhoneNo LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'),

CONSTRAINT CK\_MaleFemale CHECK ([Gender]= 'M' OR [Gender]='F'),

CONSTRAINT CK\_StateAbbr CHECK ([HomeState] LIKE '[A-Z][A-Z]'),

CONSTRAINT [PK\_Student] PRIMARY KEY CLUSTERED ([StudentID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

--Instructor Table

USE [am0742552]

GO

CREATE TABLE [dbo].[Instructor](

[InstructorID] [varchar](10) NOT NULL,

[InstructorFirstName] [varchar](50) NULL,

[InstructorLastName] [varchar](50) NULL,

[InstructorEmail] [varchar](50) NOT NULL,

[InstructorPhoneNo] [varchar](50) NULL,

[DegreeEarned] [varchar](50) NULL,

[DegreeSchool] [varchar](50) NULL,

CHECK (InstructorPhoneNo LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'),

CONSTRAINT [PK\_Instructor] PRIMARY KEY CLUSTERED

(

[InstructorID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

--Enrolled table

USE [am0742552]

GO

CREATE TABLE [dbo].[Enrolled](

[ID] [varchar](10) NOT NULL,

[InstructorID] [varchar](10) NOT NULL,

[StudentID] [varchar](10) NOT NULL,

[CourseID] [varchar](10) NOT NULL,

CONSTRAINT [PK\_Enrolled] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

--Location Table

USE [am0742552]

GO

DROP TABLE [dbo].[Location]

GO

CREATE TABLE [dbo].[Location](

[LocationID] [varchar](10) NOT NULL,

[CourseID] [varchar](10) NOT NULL,

[RoomNumber] [int] NULL,

[Building] [varchar](50) NULL,

CONSTRAINT [PK\_Location] PRIMARY KEY CLUSTERED

(

[LocationID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

--Course Table

USE [am0742552]

GO

CREATE TABLE [dbo].[Course](

[CourseID] [varchar](10) NOT NULL,

[InstructorID] [varchar](10) NOT NULL,

[ProgramID] [varchar](10) NOT NULL,

[CourseName] [varchar](50) NULL,

[ProgramCode] [char](3) NULL,

[CourseNumber] [int] NOT NULL,

[SectionNumber] [varchar](10) NOT NULL,

[CreditHours] [int] NOT NULL,

CONSTRAINT CK\_ProgramCd CHECK ([ProgramCode] LIKE '[A-Z][A-Z][A-Z]'),

CONSTRAINT CK\_CreditHours CHECK ([CreditHours] > 0),

CONSTRAINT [PK\_Course] PRIMARY KEY CLUSTERED

(

[CourseID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

--Program Table

USE [am0742552]

GO

CREATE TABLE [dbo].[Program](

[ProgramID] [varchar](10) NOT NULL,

[ProgramHead] [varchar](50) NULL,

[College] [varchar](50) NOT NULL,

[ProgramName] [varchar](50) NOT NULL,

CONSTRAINT [PK\_Program] PRIMARY KEY CLUSTERED

(

[ProgramID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

--ClassTime Table

USE [am0742552]

GO

DROP TABLE [dbo].[ClassTime]

GO

CREATE TABLE [dbo].[ClassTime](

[ClassTimeID] [varchar](10) NOT NULL,

[CourseID] [varchar](10) NOT NULL,

[DayofWeek] [varchar](50) NULL,

[ClassTime] [varchar](50) NULL,

[SemesterTaught] [varchar](50) NULL,

CONSTRAINT [PK\_ClassTime] PRIMARY KEY CLUSTERED

(

[ClassTimeID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

--Create Indices

CREATE INDEX idx\_StudentLastName ON Student(StudentLastName)

CREATE INDEX idx\_StudentPhoneNo ON Student(StudentPhoneNo)

--Foreign Keys

ALTER TABLE Enrolled ADD FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID)

ALTER TABLE Enrolled ADD FOREIGN KEY (StudentID) REFERENCES Student(StudentID)

ALTER TABLE Enrolled ADD FOREIGN KEY (CourseID) REFERENCES Course(CourseID)

ALTER TABLE Student ADD FOREIGN KEY (Program) REFERENCES Program(ProgramID)

ALTER TABLE [Location] ADD FOREIGN KEY (CourseID) REFERENCES Course(CourseID)

ALTER TABLE Course ADD FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID)

ALTER TABLE Course ADD FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID)

ALTER TABLE ClassTime ADD FOREIGN KEY (CourseID) REFERENCES Course(CourseID)

Data Insertion

-- Create 2 Programs

Insert INTO Program

Values ('000000001', 'Tom Lattimer', 'Business', 'Accounting');

Insert INTO Program

Values ('000000002', 'Frederick Durst', 'Liberal Arts', 'Anthropology');

-- Load Students

--Accounting Students

INSERT INTO Student

VALUES ('TI120448','Timmy', 'Tomms' ,'TI120448@missouristate.edu', '417-886-2325', 'MO', '01-20-1990', 'M',

146, 3.75, 'Fall 2014', '000000001');

INSERT INTO Student

VALUES ('TO855221','Tommy', 'Timms' ,'TO855221@missouristate.edu', '417-889-2523', 'MO', '11-20-1993', 'M',

31, 2.9, 'Fall 2017', '000000001');

INSERT INTO Student

VALUES ('FR987456','Freddy', 'Frederickson' ,'FR987456@missouristate.edu', '417-886-1337', 'MO', '03-22-1997', 'M',

32, 3.75, 'Spring 2017', '000000001');

INSERT INTO Student

VALUES ('HA435674','Hank', 'Hankerberg' ,'HA435674@missouristate.edu', '417-848-0201', 'MO', '10-20-2003', 'M',

18, 4, 'Fall 2016', '000000001');

INSERT INTO Student

VALUES ('AM546867','Amelia', 'Amlin' ,'AM546867@missouristate.edu', '417-889-2130', 'MO', '08-14-1999', 'F',

16, 3.75, 'Fall 2014', '000000001');

--Anthropology Students

INSERT INTO Student

VALUES ('VI485736','Virginia', 'Green' ,'VI485736@missouristate.edu', '417-848-2000', 'MO', '01-01-2000', 'F',

14, 4, 'Spring 2018', '000000002');

INSERT INTO Student

VALUES ('NO958674','Nolie', 'Nurburg' ,'NO958674@missouristate.edu', '417-890-7357', 'MO', '10-11-2001', 'F',

24, 2.75, 'Spring 2018', '000000002');

INSERT INTO Student

VALUES ('PE324654','Pepper', 'Franz' ,'PE324654@missouristate.edu', '417-888-2654', 'MO', '07-07-1999', 'F',

34, 3.9, 'Spring 2018', '000000002');

INSERT INTO Student

VALUES ('ER657919','Eric', 'Sleidel' ,'ER657919@missouristate.edu', '417-848-3000', 'MO', '06-12-2000', 'M',

41, 3.1, 'Spring 2018', '000000002');

INSERT INTO Student

VALUES ('WI657687','Wilbert', 'Williams' ,'WI657687@missouristate.edu', '417-886-4567', 'MO', '01-21-2000', 'M',

16, 2.9, 'Spring 2018', '000000002');

--Load Instructors

--Accounting Instructors

Insert INTO Instructor

Values('BI765676', 'Bill', 'Simms', 'BillSimms@missouristate.edu', '417-545-3200', 'Finance Doctorate', 'University of Tulsa');

Insert INTO Instructor

Values('PA543567', 'Patricia', 'Klein', 'PatriciaKlein@missouristate.edu', '417-433-2343', 'Master of Business Administration', 'University of Missouri Kansas City');

--Anthropology Instructors

Insert INTO Instructor

Values('ER543567', 'Erwin', 'Bittly', 'ErwinBittly@missouristate.edu', '417-890-0223', 'Masters of Anthropology', 'University of Chicago');

Insert INTO Instructor

Values('JE979654', 'Jennifer', 'Hubbard', 'JenniferHubbard@missouristate.edu', '417-313-2233', 'Anthropology Doctorate', 'University of Missouri');

--Load Courses

--12 courses for 6 classes of 2 sections each

--3 Accounting classes, 6 sections

Insert INTO Course

Values('00001', 'BI765676', '000000001', 'Introduction to Accounting', 'ACC', 101, 001, 3);

Insert INTO Course

Values('00002', 'BI765676', '000000001', 'Introduction to Accounting', 'ACC', 101, 002, 3);

Insert INTO Course

Values('00003', 'PA543567', '000000001', 'Statistical Accounting', 'ACC', 250, 001, 3);

Insert INTO Course

Values('00004', 'PA543567', '000000001', 'Statistical Accounting', 'ACC', 250, 002, 3);

Insert INTO Course

Values('00005', 'PA543567', '000000001', 'Checks and Balances', 'ACC', 330, 001, 5);

Insert INTO Course

Values('00006', 'BI765676', '000000001', 'Checks and Balances', 'ACC', 330, 002, 5);

--3 Anthropology classes, 6 sections

Insert INTO Course

Values('00007', 'ER543567', '000000002', 'Introduction to Anthopology', 'ANT', 130, 001, 3);

Insert INTO Course

Values('00008', 'ER543567', '000000002', 'Introduction to Anthopology', 'ANT', 130, 002, 3);

Insert INTO Course

Values('00009', 'JE979654', '000000002', 'Objects and Value', 'ANT', 230, 001, 4);

Insert INTO Course

Values('00010', 'JE979654', '000000002', 'Objects and Value', 'ANT', 230, 002, 4);

Insert INTO Course

Values('00011', 'JE979654', '000000002', 'Cultural Relativism', 'ANT', 350, 001, 5);

Insert INTO Course

Values('00012', 'ER543567', '000000002', 'Cultural Relativism', 'ANT', 350, 002, 5);

--Insert Locations

--Business Locations

INSERT INTO Location

values ('10001', '00001', '101', 'Blair-Shannon');

INSERT INTO Location

values ('10002', '00002', '101', 'Blair-Shannon');

INSERT INTO Location

values ('10003', '00003', '102', 'Blair-Shannon');

INSERT INTO Location

values ('10004', '00004', '102', 'Blair-Shannon');

INSERT INTO Location

values ('10005', '00005', '203', 'Blair-Shannon');

INSERT INTO Location

values ('10006', '00006', '203', 'Blair-Shannon');

--Liberal Arts Locations

INSERT INTO Location

values ('20001', '00007', '101', 'Strong Hall');

INSERT INTO Location

values ('20002', '00008', '101', 'Strong Hall');

INSERT INTO Location

values ('20003', '00009', '103', 'Strong Hall');

INSERT INTO Location

values ('20004', '00010', '103', 'Strong Hall');

INSERT INTO Location

values ('20005', '00011', '202', 'Strong Hall');

INSERT INTO Location

values ('20006', '00012', '202', 'Strong Hall');

--Load ClassTimes

--Business ClassTimes

Insert INTO ClassTime

Values('000001', '00001', 'MWF', '08:00-08:50', 'Spring 2020');

Insert INTO ClassTime

Values('000002', '00002', 'TR', '08:00-09:30', 'Spring 2020');

Insert INTO ClassTime

Values('000003', '00003', 'MWF', '10:00-10:50', 'Spring 2020');

Insert INTO ClassTime

Values('000004', '00004', 'TR', '10:00-11:30', 'Spring 2020');

Insert INTO ClassTime

Values('000005', '00005', 'MWF', '13:00-14:30', 'Spring 2020');

Insert INTO ClassTime

Values('000006', '00006', 'TR', '13:00-14:50', 'Spring 2020');

--Anthropology ClassTimes

Insert INTO ClassTime

Values('000007', '00007', 'MWF', '08:00-08:50', 'Spring 2020');

Insert INTO ClassTime

Values('000008', '00008', 'TR', '08:00-09:30', 'Spring 2020');

Insert INTO ClassTime

Values('000009', '00009', 'MWF', '10:00-10:50', 'Spring 2020');

Insert INTO ClassTime

Values('000010', '00010', 'TR', '10:00-11:30', 'Spring 2020');

Insert INTO ClassTime

Values('000011', '00011', 'MWF', '13:00-14:30', 'Spring 2020');

Insert INTO ClassTime

Values('000012', '00012', 'TR', '13:00-14:50', 'Spring 2020');

--Insert Enrollments

--Accounting Student Enrollment

--student 1

INSERT INTO [dbo].[Enrolled]

VALUES('00001','BI765676','TI120448','00001')

INSERT INTO [dbo].[Enrolled]

VALUES('00002','PA543567','TI120448','00003')

--student 2

INSERT INTO [dbo].[Enrolled]

VALUES('00003','BI765676','TO855221','00002')

INSERT INTO [dbo].[Enrolled]

VALUES('00004','PA543567','TO855221','00004')

--student 3

INSERT INTO [dbo].[Enrolled]

VALUES('00005','BI765676','FR987456','00006')

INSERT INTO [dbo].[Enrolled]

VALUES('00006','PA543567','FR987456','00003')

--student 4

INSERT INTO [dbo].[Enrolled]

VALUES('00007','BI765676','HA435674','00001')

INSERT INTO [dbo].[Enrolled]

VALUES('00008','PA543567','HA435674','00005')

--student 5

INSERT INTO [dbo].[Enrolled]

VALUES('00009','BI765676','AM546867','00002')

INSERT INTO [dbo].[Enrolled]

VALUES('00010','PA543567','AM546867','00004')

--Anthropology Student Enrollment

--student 6

INSERT INTO [dbo].[Enrolled]

VALUES('00011','ER543567','VI485736','00008')

INSERT INTO [dbo].[Enrolled]

VALUES('00012','JE979654','VI485736','00009')

--student 7

INSERT INTO [dbo].[Enrolled]

VALUES('00013','ER543567','NO958674','00012')

INSERT INTO [dbo].[Enrolled]

VALUES('00014','JE979654','NO958674','00010')

--student 8

INSERT INTO [dbo].[Enrolled]

VALUES('00015','ER543567','PE324654','00007')

INSERT INTO [dbo].[Enrolled]

VALUES('00016','JE979654','PE324654','00009')

--student 9

INSERT INTO [dbo].[Enrolled]

VALUES('00017','ER543567','ER657919','00012')

INSERT INTO [dbo].[Enrolled]

VALUES('00018','JE979654','ER657919','00010')

--student 10

INSERT INTO [dbo].[Enrolled]

VALUES('00019','ER543567','WI657687','00007')

INSERT INTO [dbo].[Enrolled]

VALUES('00020','JE979654','WI657687','00012')

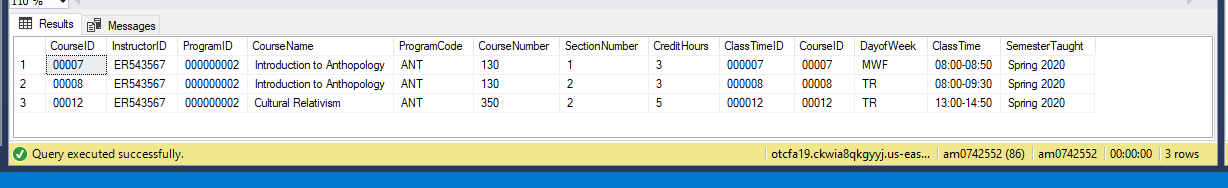
Queries

1. Pick an instructor – What classes is that instructor teaching this semester?

Paragraph Write Up:

Used a simple Select \* statement as presented on page 91 of the textbook. To select a single teacher a conditional was used on their InstructorID. This was ran on the courses table to retrieve the info about the courses the instructor is teaching with a join on ClassTime and extra conditional for SemesterTaught to select only courses taught in a specific semester.

Screenshot:



Query Used to Get Results:

SELECT \* FROM Course

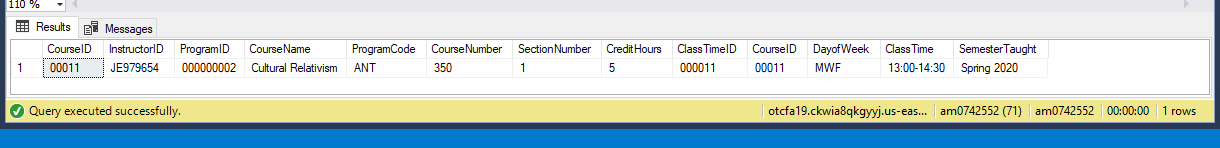
INNER JOIN ClassTime on Course.CourseID = ClassTime.CourseID

WHERE InstructorID = 'ER543567' AND ClassTime.SemesterTaught = 'Spring 2020'

Number of Rows Returned: 3

1. Pick a class – What days of the week (and at what times) is that particular class taught during a semester?

Paragraph Write Up: Used a SELECT \* statement on the Course table with and INNER JOIN statement(as described on page 127 of the text) connecting to the ClassTime table to Course Table. A conditional statement was included on CourseID to retrieve only information for a single class. The required information could have been obtained without a join and only selecting directly from the ClassTime table – but I wanted the user to be able to see the course name and information rather than just a CourseID.

Screenshot:

Query Used to Get Results: SELECT \* FROM COURSE

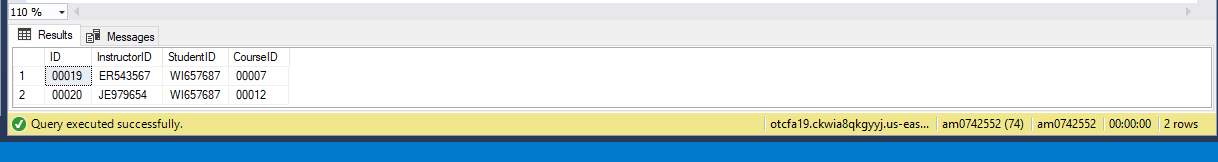
INNER JOIN ClassTime on Course.CourseID = ClassTime.CourseID

WHERE Course.CourseID = '00011';

Number of Rows Returned: 1

1. Pick a student – What classes is this student enrolled in this semester?

Paragraph Write Up: Select statement on the Enrolled Table with a conditional WHERE clause (the syntax of which is described on page 87 of the textbook) on the StudentID to only return enrollments for a single student

Screenshot:

Query Used to Get Results:

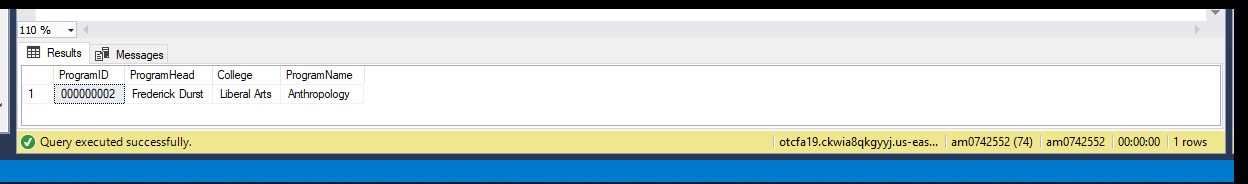
Select \* from Enrolled

Where StudentID = 'WI657687'

Number of Rows Returned: 2

1. Pick a program head – Which programs fall under that program head?

Paragraph Write Up: Created a SELECT statement to run against the Program table. Used a conditional WHERE(syntax of which came from page 87 of the text) to return only Programs under the person.

Screenshot:

Query Used to Get Results:

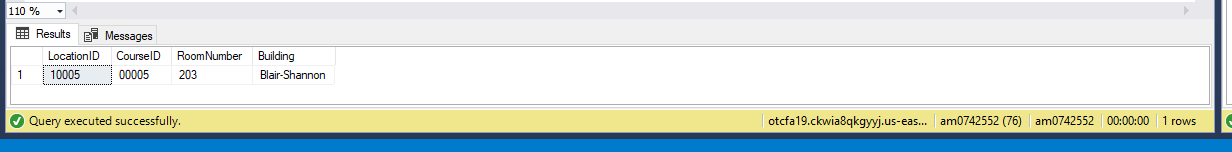
Select \* from Program

WHERE ProgramHead = 'Frederick Durst'

Number of Rows Returned: 1

1. Pick a class – Which room was this class taught in during this semester?

Paragraph Write Up:For this query I created a SELECT statement which would return all column from the Location table for rows that met the condition of having the CourseID of a specific class. I used syntax explained on page 87 of the textbook for this query.

Screenshot:

Query Used to Get Results:

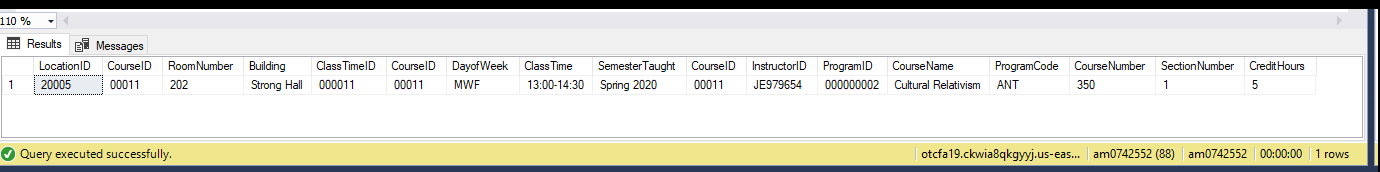
SELECT \* FROM Location

WHERE CourseID = 00005

Number of Rows Returned: 1

1. Pick a room – What classes were taught in this room on Mondays this semester? (I need to know program code, course number name and section)

Paragraph Write Up: To return the relevant information I created a Query which included a SELECT statement that joined 3 tables. I was able to join all tables together on the basis of the CourseID being present in each of the Location, ClassTime, and Course tables. To only return results for a specific classroom a WHERE statement was included to sort for entries only for a specific room number in a specific building. Additionally The DayofWeek value was compared against a string pattern containing an M followed by a wildcard operator (as detailed on page 113 of the textbook) so the query would only return classes which met on Monday. The SemesterTaught field was also compared against a string to only return courses from a specific semester.

Screenshot:

Query Used to Get Results:

SELECT \* FROM Location

INNER JOIN ClassTime ON Location.CourseID = ClassTime.CourseID

JOIN Course ON Location.CourseID = Course.CourseID

WHERE

Building = 'Strong Hall'

AND RoomNumber = 202

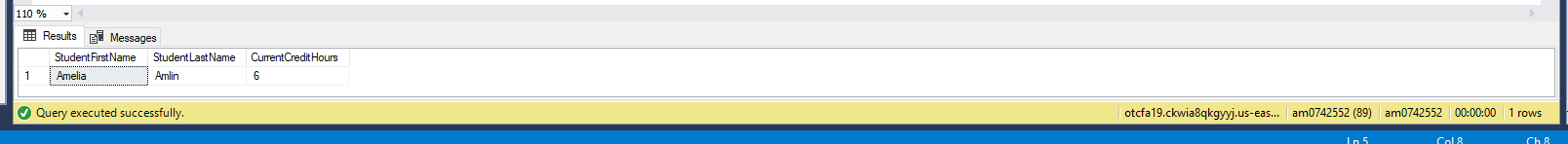
AND ClassTime.DayofWeek LIKE 'M%'

AND SemesterTaught = 'Spring 2020'

Number of Rows Returned: 1

1. Pick a student – How many credit hours is this student taking this semester?

Paragraph Write Up: To return the requested information my query included an aggregate function SUM (as described starting on page 161 of the textbook) in the SELECT statement in order to ascertain the number of credit hours the student was taking. To obtain the correct data four tables were necessary. The student table was firt needed to be able to return information on a specific student, the Enrolled table was needed to see which lcasses that student was taking, the Course table was needed to find the number of CreditHours each class the student was enrolled in had, and finally the ClassTime table was needed to be able to only include courses from the specific semester.

Screenshot:

Query Used to Get Results:

SELECT StudentFirstName, StudentLastName, SUM(CreditHours) as CurrentCreditHours FROM Student s

INNER JOIN Enrolled e on s.StudentID = e.StudentID

INNER JOIN Course c on e.CourseID = c.CourseID

INNER JOIN ClassTime ct on ct.CourseID = c.CourseID

WHERE s.StudentID = 'AM546867'

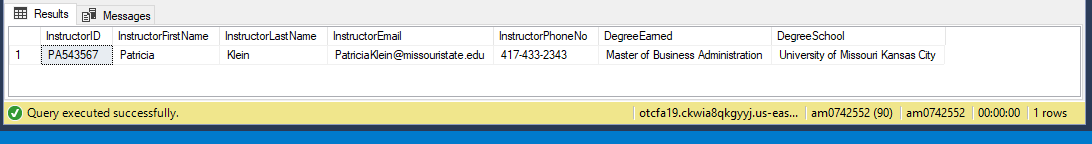
AND ct.SemesterTaught = 'Spring 2020'

GROUP BY StudentFirstName, StudentLastName

Number of Rows Returned: 1

1. Pick an instructor – What degrees does this particular instructor have and from what school?

Paragraph Write Up: To return this information my query included a SELECT statement(The syntax of which is described on page 87 of the textbook) for all columns from the Instructor table, a WHERE clause was included so only the row of the specific professor was returned.

Screenshot:

Query Used to Get Results:

SELECT \* FROM Instructor

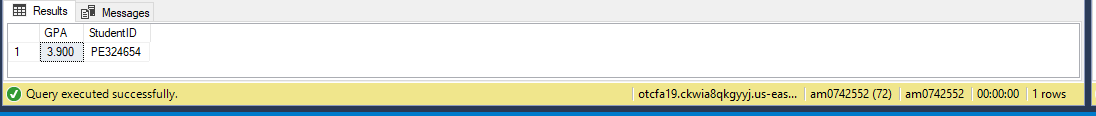
WHERE InstructorID = 'PA543567'

Number of Rows Returned: 1

1. Pick a student – What is the grade point average for all classes he/she has taken?

Paragraph Write Up: To obtain this information I created a query that selected the Student GPA and StudentID columns from the student table and included a WHERE statement on the StudentID field so the query would only return this information for a specific student.

Screenshot:



Query Used to Get Results:

SELECT Student.GPA, StudentID FROM Student

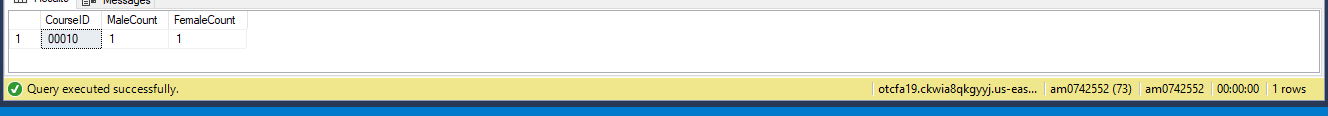
WHERE Student.StudentID = 'PE324654'

Number of Rows Returned: 1

1. Pick a class – How many males vs. females took this class this semester?

Paragraph Write Up: The query created to answer this question used a SELECT statement that called on using 3 tables joined together(syntax for joining is given on page 129 of the textbook). The CourseID was used to connect the Enrolled table with the Student table so that the gender of those enrolled in classes could be determined. The Student table was joined with the Enrolled table using StudentID to determine which students were in a specific class. By including a WHERE statement on the CourseID the query only returns information on the students enrolled in a particular class

Screenshot:



Query Used to Get Results:

SELECT Course.CourseID,

COUNT(CASE WHEN GENDER = 'M' then 1 end) as MaleCount,

COUNT(CASE WHEN GENDER = 'F' then 1 end) as FemaleCount

FROM Enrolled

JOIN Student ON Enrolled.StudentID = Student.StudentID

JOIN Course ON Enrolled.CourseID = Course.CourseID

WHERE Course.CourseID = '00010'

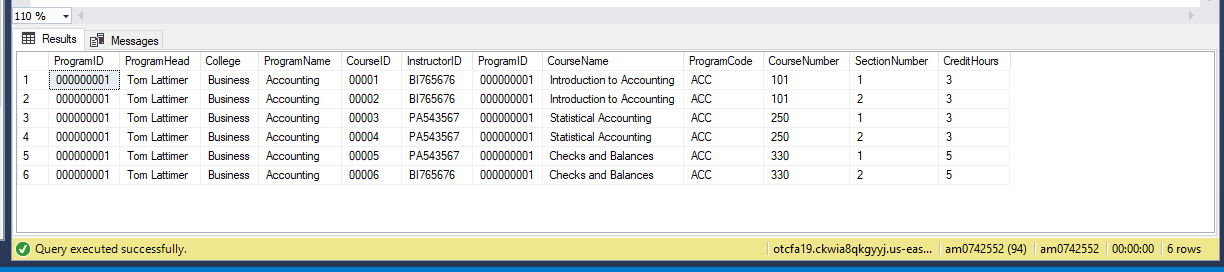
GROUP BY Course.CourseID

Number of Rows Returned: 1

1. Pick a College – Which classes are offered by all programs in this college?

Paragraph Write Up: To return this information I created a query with a SELECT statement that would return all columns from the Program table joined with columns from the Course table based on ProgramID being equal in both. Addicitonally a WHERE clause was included to return only courses from the college of interest. The syntax used for this JOIN statement was included on page 127 of the text.

Screenshot:



Query Used to Get Results:

SELECT \* FROM Program

INNER JOIN Course ON Course.ProgramID = Program.ProgramID

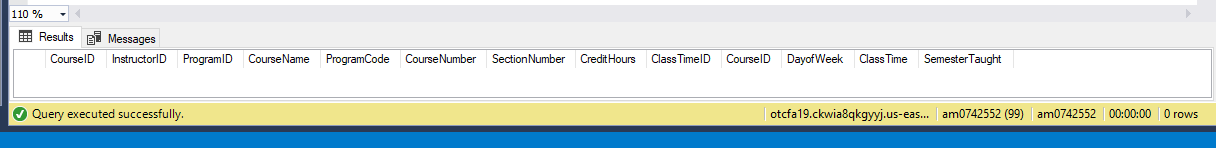
WHERE Program.College = 'Business'

Number of Rows Returned: 6

1. Write a query that shows which classes were not offered this semester (if any)

Paragraph Write Up: To return this information my query included a SELECT statement to return all columns from the Course table joined with columns from the ClassTime tables based on CourseID. A WHERE clause was included using a not equal operator (described on page 105 of the textbook) compared to the mask representing the current semester for the SemesterTaught column so only classes not taught this semester would be returned.

Screenshot:



Query Used to Get Results:

SELECT \* FROM Course

INNER JOIN ClassTime on ClassTime.CourseID = Course.CourseID

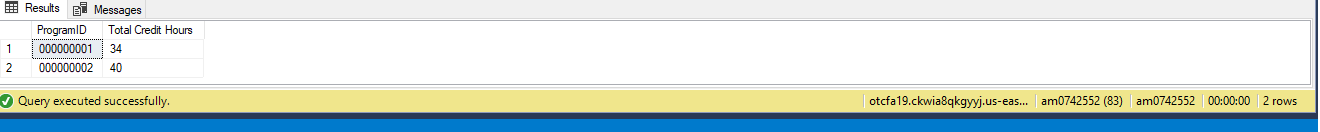
WHERE ClassTime.SemesterTaught <> 'Spring 2020'

Number of Rows Returned: 0

1. Write a query that shows how many credit hours per program were taken by all students this semester.

Paragraph Write Up: To obtain the desired information the query was created with a SELECT statement displaying the columns of ProgramID and the result of a SUM aggregate function performed on the Course CreditHours column. By joining(join syntax discussed on page 129) the Course table to the Program table on the basis of ProgramID we knew which courses were in each program. Then connecting the courses with the ClassTime table we were able to determine the credit hours of each course. By joning the enrolled table with the student table on the basis of StudentID we were able to tell who was enrolled in which course. Combining the Enrolled table with the Course table we could find the CreditHours for the courses and by finally grouping by ProgramID we were able to determine total registered hours per program. Adding a WHERE clause that filters for the current semester only is included.

Screenshot:



Query Used to Get Results:

SELECT Program.ProgramID,

SUM(CreditHours) AS TotalCreditHours

FROM Program

JOIN Course ON Course.ProgramID = Program.ProgramID

JOIN Enrolled ON Course.CourseID = Enrolled.CourseID

JOIN ClassTime ON ClassTime.CourseID = Course.CourseID

JOIN Student ON Student.StudentID = Enrolled.StudentID

WHERE ClassTime.SemesterTaught = 'Spring 2020'

GROUP BY Program.ProgramID

Number of Rows Returned: 2