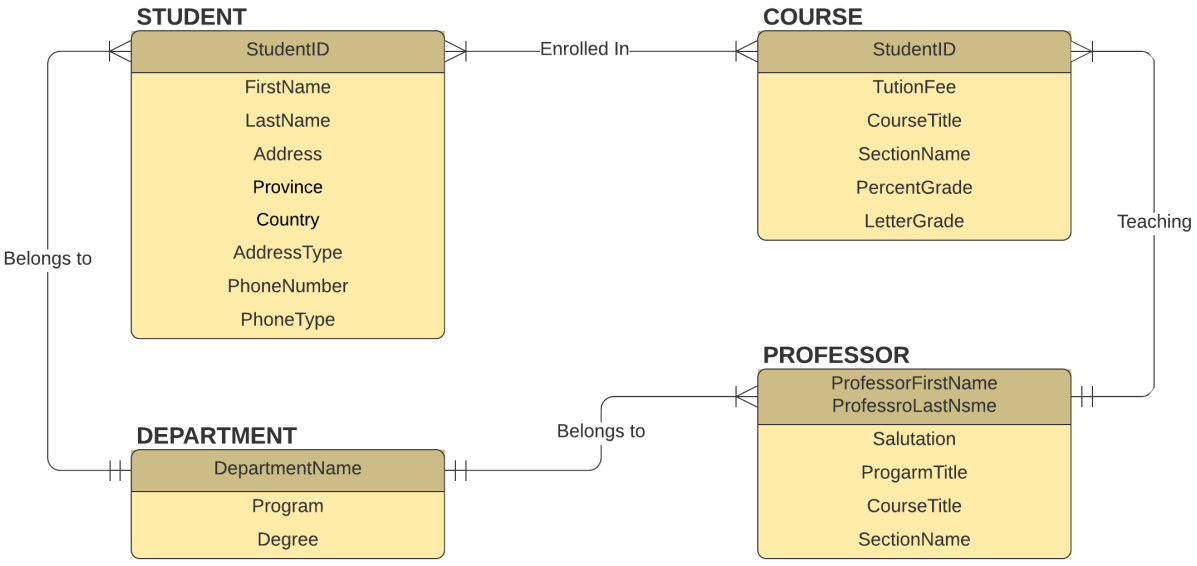
# CST 2355

# Assignment 3

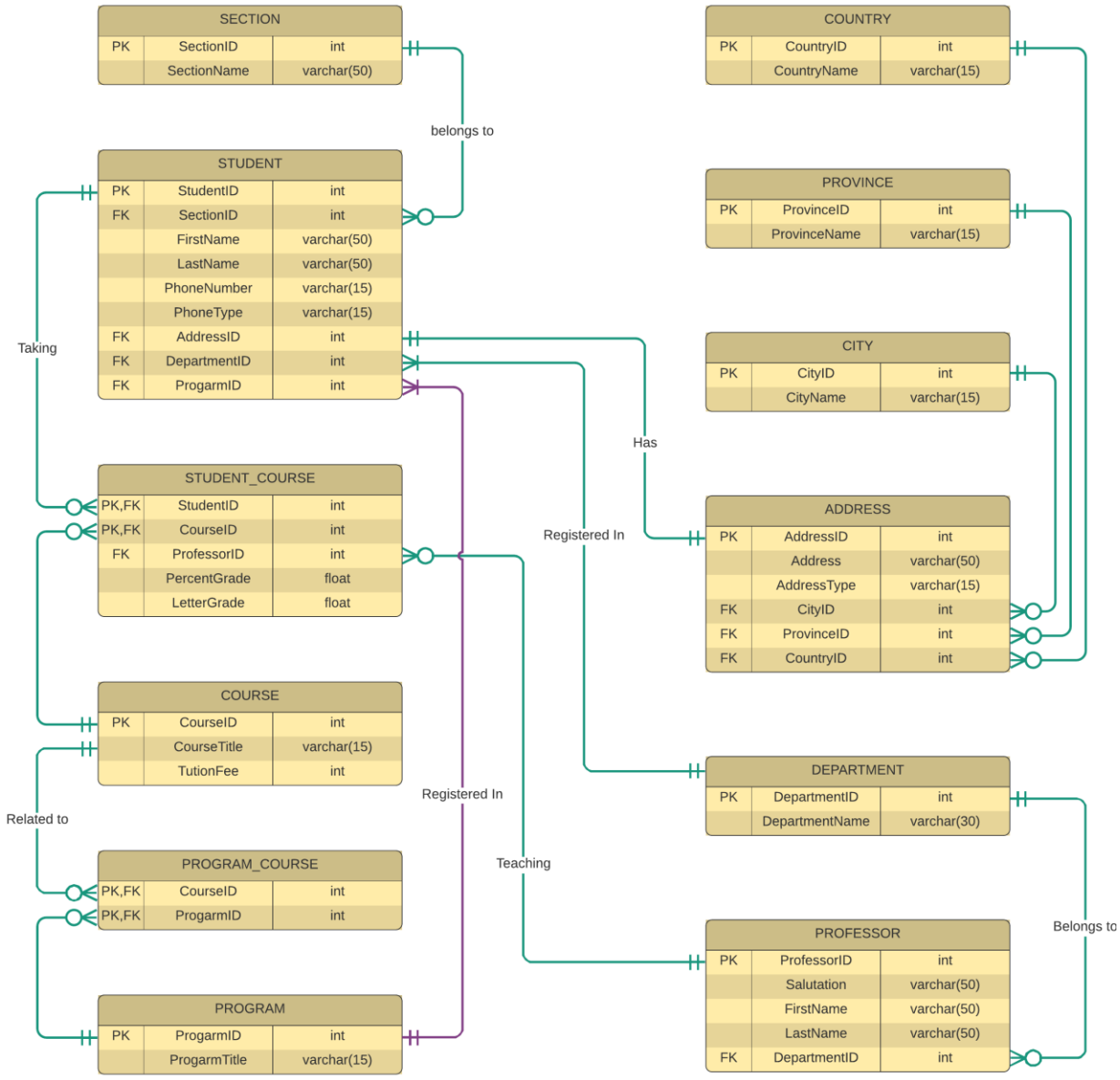

## Freddy Sourial

## 41013272


## Data Model



# Database Design



## Transformation Decisions

| ENTITY/RELATIONSHIP | TABLE/RELATIONSHIP | EXPLANATION |
|---|---|---|
| **STUDENT** | STUDENT, ADDRESS, CITY, PROVINCE, COUNTRY | Student Entity transformed to Student, Address, City, Province, Country Tables |
| **COURSE** | COURSE, SECTION, | Course Entity transformed to Course and Section Tables |
| **PROFESSOR** | PROFESSOR, PROGRAM | Professor Entity transformed to Professor and Program Tables |
| **DEPARTMENT** | DEPARTMENT | Department Entity transformed to Department Table |
| **STUDENT-COURSE** | STUDENT-COURSE, STUDENT-SECTION | Student-Course Entity Relationship transformed to Student-Course Table and Student-Section Table Relationship |
| **STUDENT-DEPARTMENT** | STUDENT-DEPARTMENT | Student-Department Entity Relationship transformed to Student-Department Table |
| **PROFESSOR-DEPARTMENT** | PROFESSOR-DEPARTMENT | Professor-Department Entity Relationship transformed to Professor-Department Table |
| **PROFESSOR-STUDENT** | PROFESSOR-STUDENT_COURSE, STUDENT-PROGRAM | Professor-Student Entity Relationship transformed to Professor-Student_Course Table, and Student-Program Table Relationship |

## Cardinality Chat

| Relationships | | | Cardinality | |
|---|---|---|---|---|
| ***Parent Entity*** | *Child Entity* | *Type* | *MIN* | *MAX* |
| **SECTION** | STUDENT | Non-identifying | M-O | 1:M |
| **STUDENT** | COURSE | Non-identifying | M-M | N:M |
| **PROGRAM** | COURSE | Non-identifying | M-M | N:M |
| **PROGRAM** | SUTDENT | Non-identifying | M-M | 1:M |
| **ADDRESS** | STUDENT | Non-identifying | M-M | 1:1 |
| **DEPARTMENT** | STUDENT | Non-identifying | M-M | 1:M |
| **DEPARTMENT** | PROFESSOR | Non-identifying | M-O | 1:M |

## Action / Trigger Chart

| Parent / Child | Operation | Action On Parent | Action On Child |
|---|---|---|---|
| **SECTION/STUDENT** | Insert | Inserted normally in Section Table | FK should be in Section Table |
| | Modify PK or FK | Can't be Modified in Section if referencing any Student table | Delete from Student if want to modify in Section Table |
| | Delete | Can't be Deleted in Section if referencing any Student table | Delete from Student if want to Delete in Section Table |
| **DEPARTMENT/ PROFESSOR** | Insert | Inserted normally in Department Table | FK should be in Department Table |
| | Modify PK or FK | Can't be Modified in Department if referencing any Professor table | Delete from Professor if want to modify in Department Table |
| | Delete | Can't be Deleted in Department if referencing any Professor table | Delete from Professor if want to Delete in Department Table |
| **COUNTRY/ ADDRESS** | Insert | Inserted normally in Country Table | FK should be in Country Table |
| | Modify PK or FK | Can't be Modified in Country if referencing any Address table | Delete from Address if want to modify in Country Table |
| | Delete | Can't be Deleted in Country if referencing any Address table | Delete from Address if want to Delete in Country Table |
| **PROVINCE/ ADDRESS** | Insert | Inserted normally in Province Table | FK should be in Province Table |
| | Modify PK or FK | Can't be Modified in Province if referencing any Address table | Delete from Address if want to modify in Province Table |
| | Delete | Can't be Deleted in Province if referencing any Address table | Delete from Address if want to Delete in Province Table |
| **ADDRESS/ STUDENT** | Insert | Inserted normally in Address Table | FK should be in Address Table |
| | Modify PK or FK | Can't be Modified in Address if referencing any Student table | Delete from Student if want to modify in Address Table |
| | Delete | Can't be Deleted in Address if referencing any Student table | Delete from Student if want to Delete in Address Table |

| | | | |
|---|---|---|---|
| **STUDENT/ STUDENT_COURSE** | Insert | Inserted normally in Student Table | FK should be in Student Table |
| | Modify PK or FK | Can't be Modified in Student if referencing any Student_Course table | Delete from Student_Course if want to modify in Student Table |
| | Delete | Can't be Deleted in Student if referencing any Student_Course table | Delete from Student_Course if want to Delete in Student Table |
| **COURSE/ STUDENT_COURSE** | Insert | Inserted normally in Course Table | FK should be in Course Table |
| | Modify PK or FK | Can't be Modified in Course if referencing any Student_Course table | Delete from Student_Course if want to modify in Course Table |
| | Delete | Can't be Deleted in Course if referencing any Student_Course table | Delete from Student_Course if want to Delete in Course Table |
| **COURSE/ PROGRAM_COURSE** | Insert | Inserted normally in Course Table | FK should be in Course Table |
| | Modify PK or FK | Can't be Modified in Course if referencing any Program_Course table | Delete from Program_Course if want to modify in Course Table |
| | Delete | Can't be Deleted in Course if referencing any Program_Course table | Delete from Program_Course if want to Delete in Course Table |
| **PROGRAM/ PROGRAM_COURSE** | Insert | Inserted normally in Program Table | FK should be in Program Table |
| | Modify PK or FK | Can't be Modified in Program if referencing any Program_Course table | Delete from Program_Course if want to modify in Program Table |
| | Delete | Can't be Deleted in Program if referencing any Program_Course table | Delete from Program_Course if want to Delete in Program Table |

## Table Charactristics

| Table | Column | Key | Not Null | Default | Data Type Oracle | Data Type SQL Server | Data Type MySQL |
|---|---|---|---|---|---|---|---|
| **STUDENT** | StudentID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | SectionID | Yes (FK) | False | Null | Int | Int | Int |
| | AddressID | Yes (FK) | False | Null | Int | Int | Int |
| | DepartmentID | Yes (FK) | False | Null | Int | Int | Int |
| | ProgramID | Yes (FK) | False | Null | Int | Int | Int |
| | FirstName | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| | LastName | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| | PhoneNumber | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| | PhoneType | No | False | Null | Varchar(50) | Varchar(50) | Varchar(50) |
| **SECTION** | SectionID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | SectionName | No | False | Null | Varchar(50) | Varchar(50) | Varchar(50) |
| **COURSE** | CourseID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | CourseTitle | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| | TutionFee | No | False | Null | Int | Int | Int |
| **PROFESSOR** | ProfessorID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | DepartmentID | Yes (FK) | False | Null | Int | Int | Int |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Salutation | No | False | Null | Varchar(50) | Varchar(50) | Varchar(50) |
| | FirstName | No | False | Null | Varchar(50) | Varchar(50) | Varchar(50) |
| | LastName | No | False | Null | Varchar(50) | Varchar(50) | Varchar(50) |
| **PROGRAM** | ProgramID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | ProgramTitle | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| **DEPARTMENT** | DepartmentID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | DepartmentName | No | False | Null | Varchar(30) | Varchar(30) | Varchar(30) |
| **ADDRESS** | AddressID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | CityID | Yes (FK) | False | Null | Int | Int | Int |
| | ProvinceID | Yes (FK) | False | Null | Int | Int | Int |
| | CountryID | Yes (FK) | False | Null | Int | Int | Int |
| | Address | No | False | Null | Varchar(50) | Varchar(50) | Varchar(50) |
| | AddressType | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| **CITY** | CityID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | CityName | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| **PROVINCE** | ProvinceID | Yes (PK) | True | Not_Null | Int | Int | Int |
| | ProvinceName | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |
| **COUNTRY** | CountryID | Yes (PK) | True | Not_Null | Int | Int | Int |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | CountryName | No | False | Null | Varchar(15) | Varchar(15) | Varchar(15) |

## Construction

*MySQL Code*

```
CREATE TABLE `COURSE` (
  `CourseID` int NOT NULL AUTO_INCREMENT,
  `CourseTitle` varchar(15),
  `TutionFee` int,
  PRIMARY KEY (`CourseID`)
);

CREATE TABLE `STUDENT` (
  `StudentID` int NOT NULL AUTO_INCREMENT,
  `SectionID` int,
  `FirstName` varchar(50),
  `LastName` varchar(50),
  `PhoneNumber` varchar(15),
  `PhoneType` varchar(15),
  `AddressID` int,
  `DepartmentID` int,
  `ProgarmID` int,
  PRIMARY KEY (`StudentID`)
);

CREATE TABLE `PROGRAM` (
  `ProgarmID` int NOT NULL AUTO_INCREMENT,
  `ProgarmTitle` varchar(15) DEFAULT 'Data Science',
```

```
  PRIMARY KEY (`ProgarmID`)
);

CREATE TABLE `PROVINCE` (
  `ProvinceID` int NOT NULL AUTO_INCREMENT,
  `ProvinceName` varchar(15),
  PRIMARY KEY (`ProvinceID`)
);

CREATE TABLE `DEPARTMENT` (
  `DepartmentID` int NOT NULL AUTO_INCREMENT,
  `DepartmentName` varchar(30),
  PRIMARY KEY (`DepartmentID`)
);

CREATE TABLE `ADDRESS` (
  `AddressID` int NOT NULL AUTO_INCREMENT,
  `Address` varchar(50),
  `AddressType` varchar(15),
  `CityID` int,
  `ProvinceID` int,
  `CountryID` int,
  PRIMARY KEY (`AddressID`)
);
```

```
CREATE TABLE `COUNTRY` (
  `CountryID` int NOT NULL AUTO_INCREMENT,
  `CountryName` varchar(15),
  PRIMARY KEY (`CountryID`)
);

CREATE TABLE `CITY` (
  `CityID` int NOT NULL AUTO_INCREMENT,
  `CityName` varchar(15),
  PRIMARY KEY (`CityID`)
);

CREATE TABLE `SECTION` (
  `SectionID` int NOT NULL AUTO_INCREMENT,
  `SectionName` varchar(50),
  PRIMARY KEY (`SectionID`)
);

CREATE TABLE `STUDENT_COURSE` (
  `StudentID` int,
  `CourseID` int,
  `ProfessorID` int,
  `PercentGrade` DECIMAL(4,1),
  `LetterGrade` DECIMAL(4,1),
  PRIMARY KEY (`StudentID`, `CourseID`)
```

```
);

CREATE TABLE `PROFESSOR` (
  `ProfessorID` int NOT NULL AUTO_INCREMENT,
  `Salutation` varchar(50) DEFAULT 'Dr',
  `FirstName` varchar(50),
  `LastName` varchar(50),
  `DepartmentID` int,
  PRIMARY KEY (`ProfessorID`)
);

CREATE TABLE `PROGRAM_COURSE` (
  `ProgarmID` int,
  `CourseID` int,
  PRIMARY KEY (`ProgarmID`, `CourseID`)
);



ALTER TABLE ADDRESS
ADD FOREIGN KEY (CityID) REFERENCES CITY(CityID),
ADD FOREIGN KEY (ProvinceID) REFERENCES PROVINCE(ProvinceID),
ADD FOREIGN KEY (CountryID) REFERENCES COUNTRY(CountryID);

ALTER TABLE STUDENT
```

```
ADD FOREIGN KEY (SectionID) REFERENCES SECTION(SectionID),

ADD FOREIGN KEY (AddressID) REFERENCES ADDRESS(AddressID),

ADD FOREIGN KEY (DepartmentID) REFERENCES
DEPARTMENT(DepartmentID),

ADD FOREIGN KEY (ProgarmID) REFERENCES PROGRAM(ProgarmID);


ALTER TABLE STUDENT_COURSE

ADD FOREIGN KEY (StudentID) REFERENCES STUDENT(StudentID),

ADD FOREIGN KEY (CourseID) REFERENCES COURSE(CourseID),

ADD FOREIGN KEY (ProfessorID) REFERENCES PROFESSOR(ProfessorID);


ALTER TABLE PROGRAM_COURSE

ADD FOREIGN KEY (ProgarmID) REFERENCES PROGRAM(ProgarmID),

ADD FOREIGN KEY (CourseID) REFERENCES COURSE(CourseID);


ALTER TABLE PROFESSOR

ADD FOREIGN KEY (DepartmentID) REFERENCES
DEPARTMENT(DepartmentID);
```

*Oracle Code*

```
CREATE TABLE COURSE (
  CourseID number(10) NOT NULL,
  CourseTitle varchar2(15),
  TutionFee number(10),
  PRIMARY KEY (CourseID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE COURSE_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER COURSE_seq_tr
 BEFORE INSERT ON COURSE FOR EACH ROW
 WHEN (NEW.CourseID IS NULL)
BEGIN
 SELECT COURSE_seq.NEXTVAL INTO :NEW.CourseID FROM DUAL;
END;
/

CREATE TABLE STUDENT (
  StudentID number(10) NOT NULL,
  SectionID number(10),
  FirstName varchar2(50),
  LastName varchar2(50),
  PhoneNumber varchar2(15),
```

```
  PhoneType varchar2(15),
  AddressID number(10),
  DepartmentID number(10),
  ProgarmID number(10),
  PRIMARY KEY (StudentID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE STUDENT_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER STUDENT_seq_tr
 BEFORE INSERT ON STUDENT FOR EACH ROW
 WHEN (NEW.StudentID IS NULL)
BEGIN
 SELECT STUDENT_seq.NEXTVAL INTO :NEW.StudentID FROM DUAL;
END;
/

CREATE TABLE PROGRAM (
  ProgarmID number(10) NOT NULL,
  ProgarmTitle varchar2(15) DEFAULT 'Data Science',
  PRIMARY KEY (ProgarmID)
);

-- Generate ID using sequence and trigger
```

```
CREATE SEQUENCE PROGRAM_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER PROGRAM_seq_tr
 BEFORE INSERT ON PROGRAM FOR EACH ROW
 WHEN (NEW.ProgarmID IS NULL)
BEGIN
 SELECT PROGRAM_seq.NEXTVAL INTO :NEW.ProgarmID FROM DUAL;
END;
/

CREATE TABLE PROVINCE (
  ProvinceID number(10) NOT NULL,
  ProvinceName varchar2(15),
  PRIMARY KEY (ProvinceID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE PROVINCE_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER PROVINCE_seq_tr
 BEFORE INSERT ON PROVINCE FOR EACH ROW
 WHEN (NEW.ProvinceID IS NULL)
BEGIN
 SELECT PROVINCE_seq.NEXTVAL INTO :NEW.ProvinceID FROM DUAL;
END;
```

```
/

CREATE TABLE DEPARTMENT (
  DepartmentID number(10) NOT NULL,
  DepartmentName varchar2(30),
  PRIMARY KEY (DepartmentID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE DEPARTMENT_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER DEPARTMENT_seq_tr
 BEFORE INSERT ON DEPARTMENT FOR EACH ROW
 WHEN (NEW.DepartmentID IS NULL)
BEGIN
 SELECT DEPARTMENT_seq.NEXTVAL INTO :NEW.DepartmentID FROM DUAL;
END;
/

CREATE TABLE ADDRESS (
  AddressID number(10) NOT NULL,
  Address varchar2(50),
  AddressType varchar2(15),
  CityID number(10),
  ProvinceID number(10),
```

```
  CountryID number(10),
  PRIMARY KEY (AddressID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE ADDRESS_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER ADDRESS_seq_tr
 BEFORE INSERT ON ADDRESS FOR EACH ROW
 WHEN (NEW.AddressID IS NULL)
BEGIN
 SELECT ADDRESS_seq.NEXTVAL INTO :NEW.AddressID FROM DUAL;
END;
/

CREATE TABLE COUNTRY (
  CountryID number(10) NOT NULL,
  CountryName varchar2(15),
  PRIMARY KEY (CountryID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE COUNTRY_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER COUNTRY_seq_tr
```

```
 BEFORE INSERT ON COUNTRY FOR EACH ROW
 WHEN (NEW.CountryID IS NULL)
BEGIN
 SELECT COUNTRY_seq.NEXTVAL INTO :NEW.CountryID FROM DUAL;
END;
/

CREATE TABLE CITY (
  CityID number(10) NOT NULL,
  CityName varchar2(15),
  PRIMARY KEY (CityID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE CITY_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER CITY_seq_tr
 BEFORE INSERT ON CITY FOR EACH ROW
 WHEN (NEW.CityID IS NULL)
BEGIN
 SELECT CITY_seq.NEXTVAL INTO :NEW.CityID FROM DUAL;
END;
/

CREATE TABLE SECTION (
```

```
  SectionID number(10) NOT NULL,
  SectionName varchar2(50),
  PRIMARY KEY (SectionID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE SECTION_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER SECTION_seq_tr
 BEFORE INSERT ON SECTION FOR EACH ROW
 WHEN (NEW.SectionID IS NULL)
BEGIN
 SELECT SECTION_seq.NEXTVAL INTO :NEW.SectionID FROM DUAL;
END;
/

CREATE TABLE STUDENT_COURSE (
  StudentID number(10),
  CourseID number(10),
  ProfessorID number(10),
  PercentGrade NUMBER(4,1),
  LetterGrade NUMBER(4,1),
  PRIMARY KEY (StudentID, CourseID)
);
```

```
CREATE TABLE PROFESSOR (
  ProfessorID number(10) NOT NULL,
  Salutation varchar2(50) DEFAULT 'Dr',
  FirstName varchar2(50),
  LastName varchar2(50),
  DepartmentID number(10),
  PRIMARY KEY (ProfessorID)
);

-- Generate ID using sequence and trigger
CREATE SEQUENCE PROFESSOR_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER PROFESSOR_seq_tr
 BEFORE INSERT ON PROFESSOR FOR EACH ROW
 WHEN (NEW.ProfessorID IS NULL)
BEGIN
 SELECT PROFESSOR_seq.NEXTVAL INTO :NEW.ProfessorID FROM DUAL;
END;
/

CREATE TABLE PROGRAM_COURSE (
  ProgarmID number(10),
  CourseID number(10),
  PRIMARY KEY (ProgarmID, CourseID)
);
```

```
ALTER TABLE ADDRESS
ADD FOREIGN KEY (CityID) REFERENCES CITY(CityID),
ADD FOREIGN KEY (ProvinceID) REFERENCES PROVINCE(ProvinceID),
ADD FOREIGN KEY (CountryID) REFERENCES COUNTRY(CountryID);

ALTER TABLE STUDENT
ADD FOREIGN KEY (SectionID) REFERENCES SECTION(SectionID),
ADD FOREIGN KEY (AddressID) REFERENCES ADDRESS(AddressID),
ADD FOREIGN KEY (DepartmentID) REFERENCES
DEPARTMENT(DepartmentID),
ADD FOREIGN KEY (ProgarmID) REFERENCES PROGRAM(ProgarmID);

ALTER TABLE STUDENT_COURSE
ADD FOREIGN KEY (StudentID) REFERENCES STUDENT(StudentID),
ADD FOREIGN KEY (CourseID) REFERENCES COURSE(CourseID),
ADD FOREIGN KEY (ProfessorID) REFERENCES PROFESSOR(ProfessorID);

ALTER TABLE PROGRAM_COURSE
ADD FOREIGN KEY (ProgarmID) REFERENCES PROGRAM(ProgarmID),
ADD FOREIGN KEY (CourseID) REFERENCES COURSE(CourseID);

ALTER TABLE PROFESSOR
```

```
ADD FOREIGN KEY (DepartmentID) REFERENCES
DEPARTMENT(DepartmentID);
```

*MS SQL Server Code*

```
CREATE TABLE COURSE (
  [CourseID] int NOT NULL IDENTITY,
  [CourseTitle] varchar(15),
  [TutionFee] int,
  PRIMARY KEY ([CourseID])
);

CREATE TABLE STUDENT (
  [StudentID] int NOT NULL IDENTITY,
  [SectionID] int,
  [FirstName] varchar(50),
  [LastName] varchar(50),
  [PhoneNumber] varchar(15),
  [PhoneType] varchar(15),
  [AddressID] int,
  [DepartmentID] int,
  [ProgarmID] int,
  PRIMARY KEY ([StudentID])
);

CREATE TABLE PROGRAM (
  [ProgarmID] int NOT NULL IDENTITY,
```

```
  [ProgarmTitle] varchar(15) DEFAULT 'Data Science',
  PRIMARY KEY ([ProgarmID])
);

CREATE TABLE PROVINCE (
  [ProvinceID] int NOT NULL IDENTITY,
  [ProvinceName] varchar(15),
  PRIMARY KEY ([ProvinceID])
);

CREATE TABLE DEPARTMENT (
  [DepartmentID] int NOT NULL IDENTITY,
  [DepartmentName] varchar(30),
  PRIMARY KEY ([DepartmentID])
);

CREATE TABLE ADDRESS (
  [AddressID] int NOT NULL IDENTITY,
  [Address] varchar(50),
  [AddressType] varchar(15),
  [CityID] int,
  [ProvinceID] int,
  [CountryID] int,
```

```
  PRIMARY KEY ([AddressID])
);

CREATE TABLE COUNTRY (
  [CountryID] int NOT NULL IDENTITY,
  [CountryName] varchar(15),
  PRIMARY KEY ([CountryID])
);

CREATE TABLE CITY (
  [CityID] int NOT NULL IDENTITY,
  [CityName] varchar(15),
  PRIMARY KEY ([CityID])
);

CREATE TABLE SECTION (
  [SectionID] int NOT NULL IDENTITY,
  [SectionName] varchar(50),
  PRIMARY KEY ([SectionID])
);

CREATE TABLE STUDENT_COURSE (
  [StudentID] int,
```

```
  [CourseID] int,
  [ProfessorID] int,
  [PercentGrade] DECIMAL(4,1),
  [LetterGrade] DECIMAL(4,1),
  PRIMARY KEY ([StudentID], [CourseID])
);

CREATE TABLE PROFESSOR (
  [ProfessorID] int NOT NULL IDENTITY,
  [Salutation] varchar(50) DEFAULT 'Dr',
  [FirstName] varchar(50),
  [LastName] varchar(50),
  [DepartmentID] int,
  PRIMARY KEY ([ProfessorID])
);

CREATE TABLE PROGRAM_COURSE (
  [ProgarmID] int,
  [CourseID] int,
  PRIMARY KEY ([ProgarmID], [CourseID])
);
```

```
ALTER TABLE ADDRESS

ADD FOREIGN KEY (CityID) REFERENCES CITY(CityID);

ALTER TABLE ADDRESS

ADD FOREIGN KEY (ProvinceID) REFERENCES
dbo.PROVINCE(ProvinceID);

ALTER TABLE ADDRESS

ADD FOREIGN KEY (CountryID) REFERENCES dbo.COUNTRY(CountryID);


ALTER TABLE STUDENT

ADD FOREIGN KEY (SectionID) REFERENCES SECTION(SectionID);

ALTER TABLE STUDENT

ADD FOREIGN KEY (AddressID) REFERENCES dbo.ADDRESS(AddressID);

ALTER TABLE STUDENT

ADD FOREIGN KEY (DepartmentID) REFERENCES
dbo.DEPARTMENT(DepartmentID);

ALTER TABLE STUDENT

ADD FOREIGN KEY (ProgarmID) REFERENCES dbo.PROGRAM(ProgarmID);


ALTER TABLE STUDENT_COURSE

ADD FOREIGN KEY (StudentID) REFERENCES STUDENT(StudentID);

ALTER TABLE STUDENT_COURSE

ADD FOREIGN KEY (CourseID) REFERENCES dbo.COURSE(CourseID);

ALTER TABLE STUDENT_COURSE
```

```
ADD FOREIGN KEY (ProfessorID) REFERENCES
dbo.PROFESSOR(ProfessorID);


ALTER TABLE PROGRAM_COURSE

ADD FOREIGN KEY (ProgarmID) REFERENCES PROGRAM(ProgarmID);

ALTER TABLE PROGRAM_COURSE

ADD FOREIGN KEY (CourseID) REFERENCES dbo.COURSE(CourseID);


ALTER TABLE PROFESSOR

ADD FOREIGN KEY (DepartmentID) REFERENCES
DEPARTMENT(DepartmentID);
```

# Reports

## *Program Enrollment Report*

MySQL Query

```
SELECT p.ProgarmTitle,COUNT(s.StudentID) as NumberOfStudents
from program p inner JOIN student s on p.ProgarmID=s.ProgarmID

GROUP by p.ProgarmTitle
```

| ProgarmTitle | NumberOfStudents |
|---|---|
| BS | 5 |

Same Query will work on MS SQL Server and Oracle

## *Course Enrollment Report*

MySQL Query

```
SELECT p.ProgarmTitle, c.CourseTitle,COUNT(s.StudentID) as
NumberOfStudents,c.TutionFee from student s inner join

PROGRAM p on s.ProgarmID=p.ProgarmID

LEFT JOIN student_course sc on s.StudentID=sc.StudentID

RIGHT JOIN course c on sc.CourseID=c.CourseID

GROUP by c.CourseTitle,c.TutionFee,p.ProgarmTitle
```

| ProgarmTitle | CourseTitle | NumberOfStudents | TutionFee |
|---|---|---|---|
| BS | Applied Physics | 2 | 30 |
| *NULL* | Computer Networ | 0 | 50 |
| BS | Data Structures | 2 | 35 |
| *NULL* | Design & Analys | 0 | 40 |
| BS | Discrete Struct | 1 | 65 |
| BS | Introduction to | 2 | 50 |
| *NULL* | Linear Algebra | 0 | 60 |
| *NULL* | Operating Syste | 0 | 55 |
| BS | Programming Fun | 2 | 50 |

Same Query will work on MS SQL Server and Oracle

*Program/Course Enrollment Report*

MySQL Query

```
SELECT p.ProgarmTitle, c.CourseTitle,COUNT(s.StudentID) as
NumberOfStudents,SUM(c.TutionFee) TotalTution from student s

inner join PROGRAM p on s.ProgarmID=p.ProgarmID

LEFT JOIN student_course sc on s.StudentID=sc.StudentID

RIGHT JOIN course c on sc.CourseID=c.CourseID

GROUP by c.CourseTitle,p.ProgarmTitle
```

| ProgarmTitle | CourseTitle | NumberOfStudents | TotalTution |
|---|---|---|---|
| BS | Applied Physics | 2 | 60 |
| *NULL* | Computer Networ | 0 | 50 |
| BS | Data Structures | 2 | 70 |
| *NULL* | Design & Analys | 0 | 40 |
| BS | Discrete Struct | 1 | 65 |
| BS | Introduction to | 2 | 100 |
| *NULL* | Linear Algebra | 0 | 60 |
| *NULL* | Operating Syste | 0 | 55 |
| BS | Programming Fun | 2 | 100 |

Same Query will work on MS SQL Server and Oracle

*Course/Program Matrix*

MySQL Query

```
SELECT c.CourseTitle,p.ProgarmTitle from program p

INNER join program_course pc

on p.ProgarmID=pc.ProgarmID

INNER JOIN course c

on pc.CourseID=c.CourseID
```

*Student Grade Average*

MySQL Query

```
SELECT CONCAT(s.FirstName,s.LastName) as StudentName
,COUNT(c.CourseID) NumberOfCourses,AVG(sc.PercentGrade) as
GradesAverage,c.TutionFee from student s inner join

PROGRAM p on s.ProgarmID=p.ProgarmID

LEFT JOIN student_course sc on s.StudentID=sc.StudentID

RIGHT JOIN course c on sc.CourseID=c.CourseID

GROUP by StudentName
```

| StudentName | NumberOfCourses | GradesAverage | TutionFee |
|---|---|---|---|
| *NULL* | 4 | *NULL* | 60 |
| MichealCorlone | 4 | 3.20000 | 50 |
| VitoCorlone | 5 | 3.20000 | 50 |

Same Query will work on MS SQL Server and Oracle

### *Student Professor View*

MySQL Query

```
CREATE VIEW StudentProfessorView AS

SELECT CONCAT(p.FirstName,p.LastName) as
ProfName,c.CourseTitle,CONCAT(s.FirstName,s.LastName) as
StudentName

from student s INNER JOIN student_course sc on
s.StudentID=sc.StudentID

INNER JOIN course c on sc.CourseID=c.CourseID

LEFT JOIN professor p on p.ProfessorID=sc.ProfessorID
```

| ProfName | CourseTitle | StudentName |
|---|---|---|
| MarioBadr | Introduction to | MichealCorlone |
| GaryBaumgartner | Programming Fun | MichealCorlone |
| JenniferCampbell | Applied Physics | MichealCorlone |
| JenniferCampbell | Data Structures | MichealCorlone |
| GaryBaumgartner | Introduction to | VitoCorlone |
| MichelleWahl Craig | Programming Fun | VitoCorlone |
| SteveEngels | Applied Physics | VitoCorlone |
| MarioBadr | Data Structures | VitoCorlone |
| GaryBaumgartner | Discrete Struct | VitoCorlone |

Same Query will work on MS SQL Server and Oracle

## Requests

*Update the PhoneType's default to "not provided". Fix the PhoneTypes that are blank that have Phone numbers and don't have Phone numbers.*

```
ALTER TABLE STUDENT ALTER PhoneType SET DEFAULT 'Not Provided';

UPDATE STUDENT SET PhoneType='Not Provided' WHERE
PhoneType=null;
```

*Change the Section's default to 450. No sections can be under 450. Fix the data.*

```
ALTER TABLE STUDENT ALTER SectionID SET DEFAULT 450;
```

*Change TuitionAmount so it has to digits after the decimal. (56 becomes 56.00)*

```
ALTER TABLE COURSE MODIFY TutionFee DECIMAL(4,2)
```