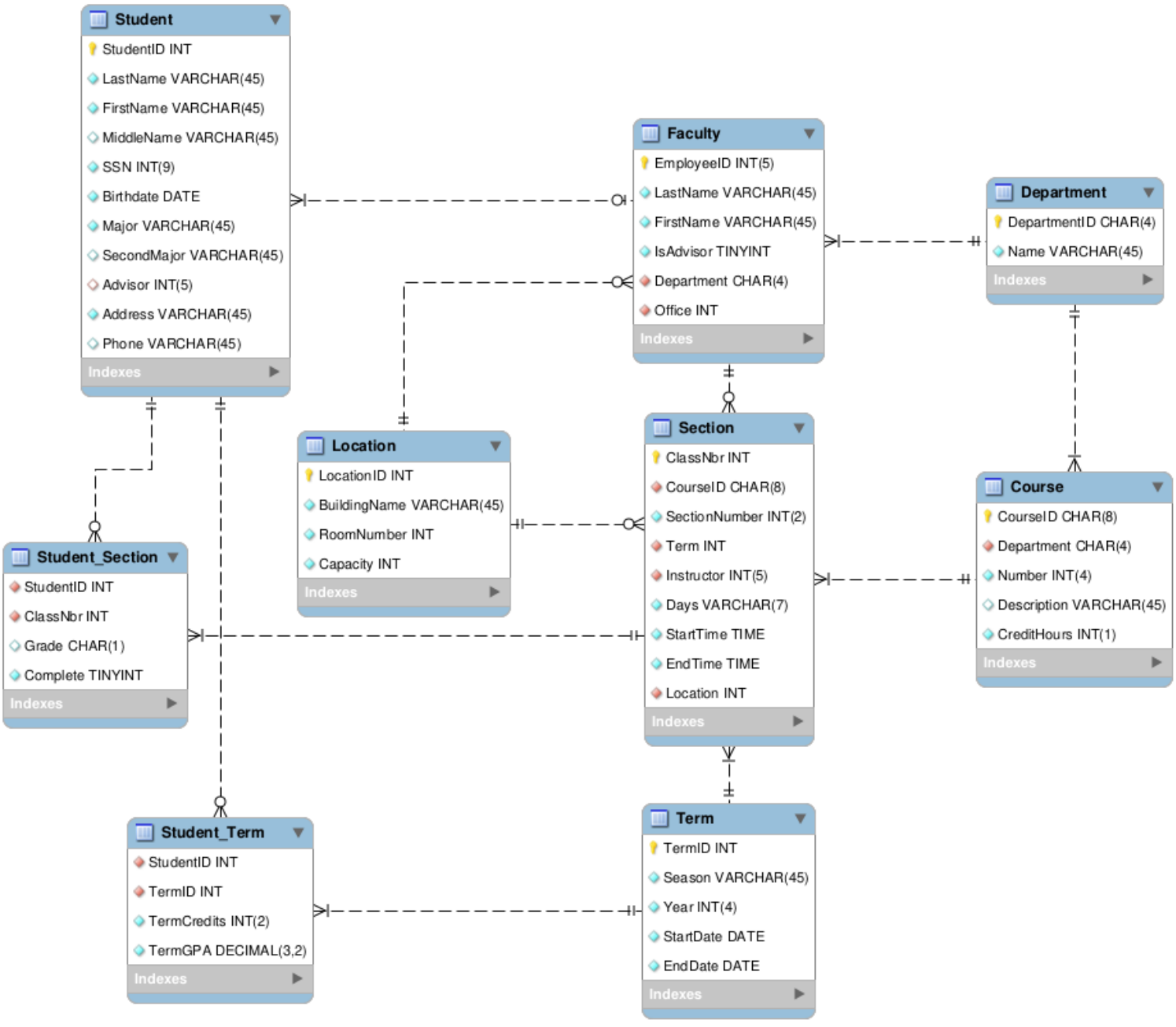
Chris Scarola
CSCI 3287

Project 1 – City College Registration



DDL code:

```
-- -----------------------------------------------------
-- Schema CCRegistration
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `CCRegistration` DEFAULT CHARACTER SET utf8 ;
USE `CCRegistration` ;

-- -----------------------------------------------------
-- Table `CCRegistration`.`Department`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Department` (
  `DepartmentID` CHAR(4) NOT NULL,
  `Name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`DepartmentID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Location`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Location` (
  `LocationID` INT NOT NULL,
  `BuildingName` VARCHAR(45) NOT NULL,
  `RoomNumber` INT NOT NULL,
  `Capacity` INT NOT NULL,
  PRIMARY KEY (`LocationID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Faculty`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Faculty` (
  `EmployeeID` INT(5) NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `FirstName` VARCHAR(45) NOT NULL,
  `IsAdvisor` TINYINT NOT NULL DEFAULT 0,
  `Department` CHAR(4) NOT NULL,
  `Office` INT NOT NULL,
  PRIMARY KEY (`EmployeeID`),
  INDEX `DepartmentID_idx` (`Department` ASC),
  INDEX `LocationID_idx` (`Office` ASC),
  CONSTRAINT `DepartmentID`
    FOREIGN KEY (`Department`)
    REFERENCES `CCRegistration`.`Department` (`DepartmentID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `LocationID`
```

```
    FOREIGN KEY (`Office`)
    REFERENCES `CCRegistration`.`Location` (`LocationID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Student`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Student` (
  `StudentID` INT NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `FirstName` VARCHAR(45) NOT NULL,
  `MiddleName` VARCHAR(45) NULL,
  `SSN` INT(9) NOT NULL,
  `Birthdate` DATE NOT NULL,
  `Major` VARCHAR(45) NOT NULL,
  `SecondMajor` VARCHAR(45) NULL,
  `Advisor` INT(5) NULL,
  `Address` VARCHAR(45) NOT NULL,
  `Phone` VARCHAR(45) NULL,
  PRIMARY KEY (`StudentID`),
  INDEX `EmployeeID_idx` (`Advisor` ASC),
  INDEX `Last_first` (`LastName` ASC, `FirstName` ASC),
  INDEX `First_last` (`FirstName` ASC, `LastName` ASC),
  CONSTRAINT `EmployeeID`
    FOREIGN KEY (`Advisor`)
    REFERENCES `CCRegistration`.`Faculty` (`EmployeeID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Term`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Term` (
  `Year` INT(4) NOT NULL,
  `Season` VARCHAR(45) NOT NULL,
  `StartDate` DATE NOT NULL,
  `EndDate` DATE NOT NULL,
  CONSTRAINT `TermID` PRIMARY KEY (`Year`,`Season`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Course`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Course` (
  `Department` CHAR(4) NOT NULL,
  `Number` INT(4) NOT NULL,
  `Description` VARCHAR(45) NULL,
  `CreditHours` INT(1) NOT NULL,
  CONSTRAINT `CourseID` PRIMARY KEY (`Department`,`Number`),
  INDEX `DepartmentID_idx` (`Department` ASC),
  CONSTRAINT `DepartmentID`
    FOREIGN KEY (`Department`)
    REFERENCES `CCRegistration`.`Department` (`DepartmentID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `TermID`
    FOREIGN KEY (`TermID`)
    REFERENCES `CCRegistration`.`Term` (`TermID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
  CHECK (CreditHours >= 1 AND CreditHours <= 4))
ENGINE = InnoDB
PACK_KEYS = DEFAULT;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Section`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Section` (
  `CourseID` CHAR(8) NOT NULL,
  `SectionNumber` INT(2) NOT NULL,
  `Term` INT NOT NULL,
  `Instructor` INT(5) NOT NULL,
  `Days` VARCHAR(7) NOT NULL,
  `StartTime` TIME NOT NULL,
  `EndTime` TIME NOT NULL,
  `Location` INT NOT NULL,
  CONSTRAINT `ClassNbr` PRIMARY KEY (`Term`,`CourseID`,`SectionNumber`),
  INDEX `CourseID_idx` (`CourseID` ASC),
  INDEX `EmployeeID_idx` (`Instructor` ASC),
  INDEX `LocationID_idx` (`Location` ASC),
  INDEX `TermID_idx` (`Term` ASC),
  INDEX `Dept_Term` (`CourseID` ASC, `Term` ASC),
  UNIQUE INDEX `SectionNumber_UNIQUE` (`SectionNumber` ASC),
  CONSTRAINT `CourseID`
    FOREIGN KEY (`CourseID`)
    REFERENCES `CCRegistration`.`Course` (`CourseID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `EmployeeID`
    FOREIGN KEY (`Instructor`)
    REFERENCES `CCRegistration`.`Faculty` (`EmployeeID`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `LocationID`
    FOREIGN KEY (`Location`)
    REFERENCES `CCRegistration`.`Location` (`LocationID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
  CHECK (SectionNumber >= 1 AND SectionNumber <= 12))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Student_Section`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Student_Section` (
  `StudentID` INT NOT NULL,
  `ClassNbr` INT NOT NULL,
  `Grade` CHAR(1) NULL DEFAULT NULL,
  `Complete` TINYINT NOT NULL DEFAULT 0,
  INDEX `StudentID_idx` (`StudentID` ASC),
  INDEX `ClassNbr_idx` (`ClassNbr` ASC),
  CONSTRAINT `StudentID`
    FOREIGN KEY (`StudentID`)
    REFERENCES `CCRegistration`.`Student` (`StudentID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ClassNbr`
    FOREIGN KEY (`ClassNbr`)
    REFERENCES `CCRegistration`.`Section` (`ClassNbr`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CCRegistration`.`Student_Term`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CCRegistration`.`Student_Term` (
  `StudentID` INT NOT NULL,
  `TermID` INT NOT NULL,
  `TermCredits` INT(2) NOT NULL DEFAULT 0,
  `TermGPA` DECIMAL(3,2) NOT NULL DEFAULT 0,
  INDEX `StudentID_idx` (`StudentID` ASC),
  INDEX `TermID_idx` (`TermID` ASC),
  CONSTRAINT `StudentID`
    FOREIGN KEY (`StudentID`)
    REFERENCES `CCRegistration`.`Student` (`StudentID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
```

```
  CONSTRAINT `TermID`
    FOREIGN KEY (`TermID`)
    REFERENCES `CCRegistration`.`Term` (`TermID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
  CHECK (TermCredits <= 18))
ENGINE = InnoDB;
```

Assumptions

- One of the indexes wants to reference a department for each semester. Since the courseID is the department and course number, the section table indexes on the courseID and termID so a query that uses WHERE and LIKE can be used to find the list of unique departments per semester
- The Course table is meant to act like a history of possible courses, so it is separate from the sections
- For a unique course per student, the courseID can be checked unique after joining the Section and Student tables
- The historical requirement can be met by using a query that checks the term ID using WHERE and LIKE since the termID is the Year+Season and look at the SSN instead of StudentID. Another aspect to this assumption is the possibility of the StudentID just being entered as SSNs instead of having a check every query. That way just make them the same for the old students and separate for the new ones.
- Approval is out of scope of the database, the assumption is that anything entered was already approved.
- To handle the 3 course limit for each instructor, should be application side, or based on query, by checking to see if the faculty member has < 3 courses per term in order to add.
- The overall GPA can be calculated using the most recent term grades per student by the application. The database keeps all the grades but does not replace things or calculate averages.