University Records Management

Creating, analyzing, classifying and maintaining university databases

By Marsela Aliaj, Araba Budu-Anguah, Sean Graham, Austin Winter

Overview of the topic

The data we are going to use today is all going to be fictional data.

We are going to create our own university and produce a few datasets on how we imagine this process is made in real life scenarios.

Objective s

1.Creating database that contains at least three(3) tables

2.Including in our tables a primary and/or foreign key and data types as varchar, int, decimal, etc.

3.Writing summary queries by using group by clause

4.Demonstrate different examples of procedures and functions

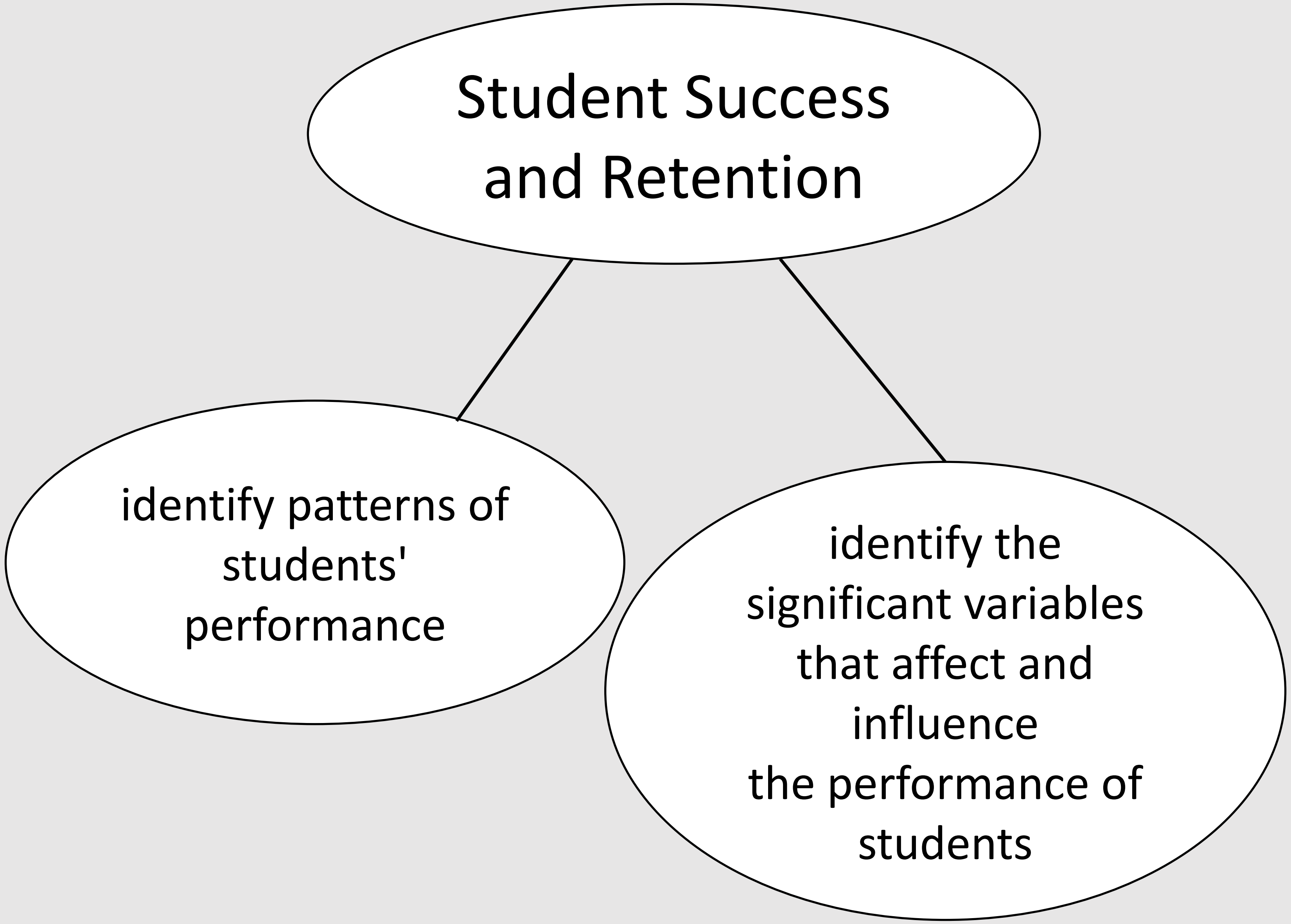
Significance of Student Records

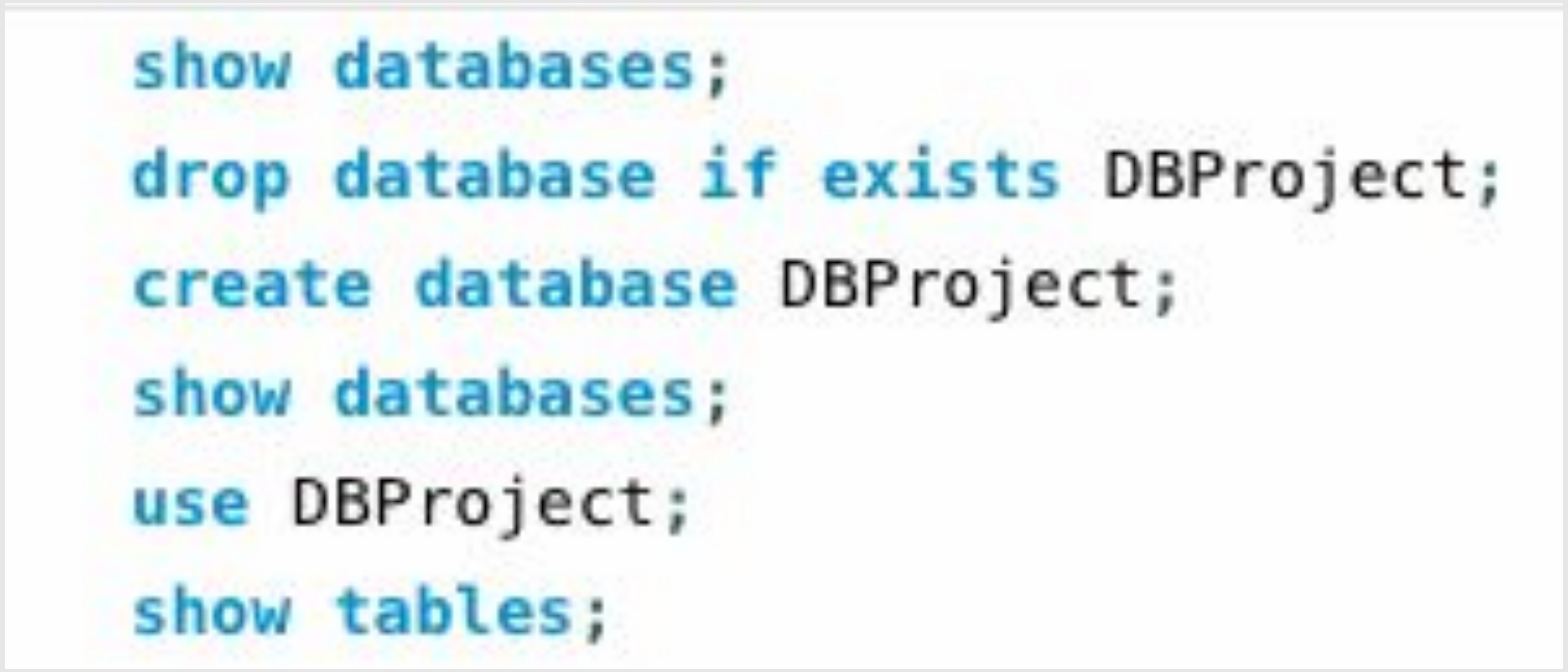
OPERATIONAL

LEGAL

EMERGENCY

FISCAL


Student Success and Retention
identify patterns of students' performance
identify the significant variables that affect and influence the performance of students

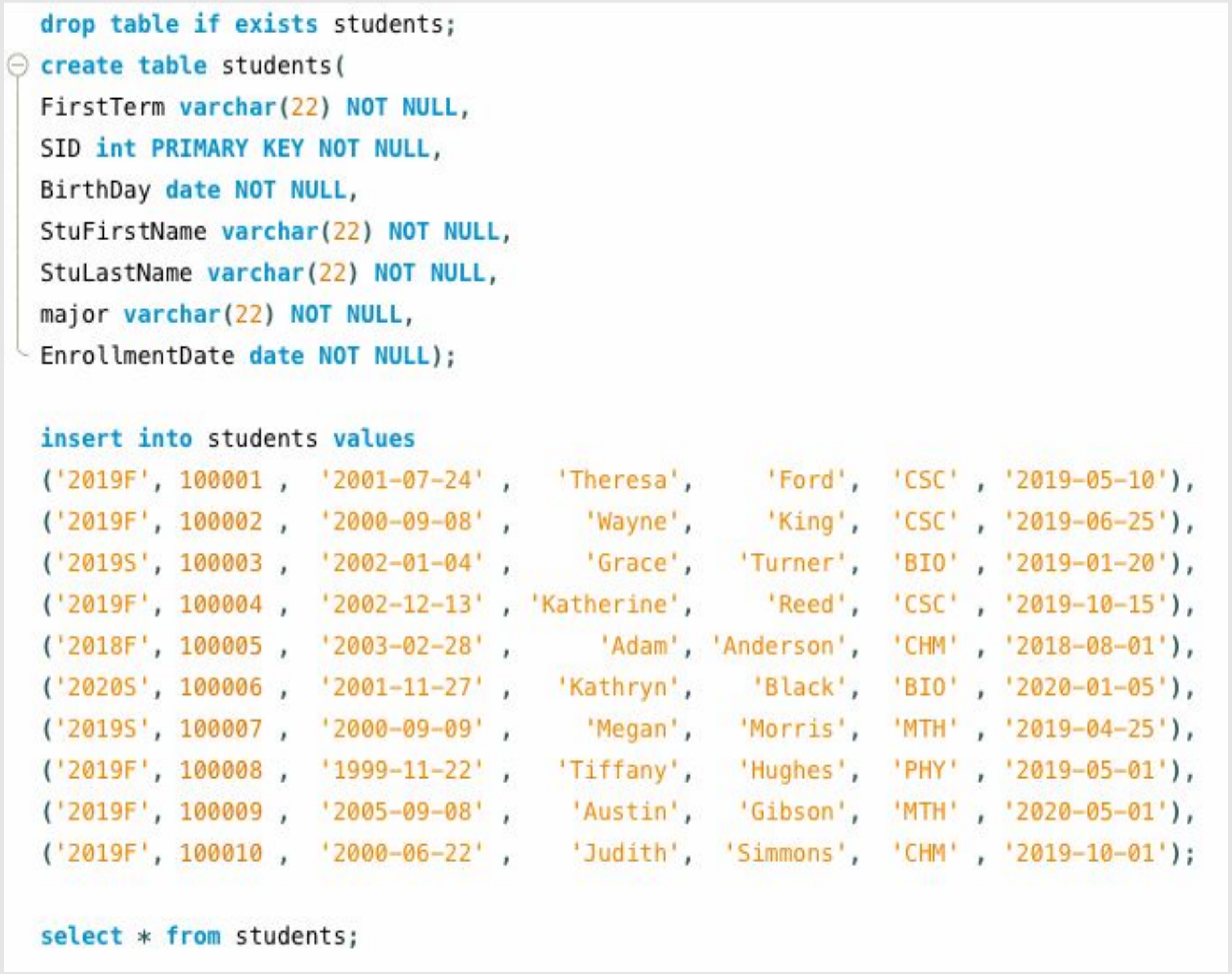


```
show databases;
drop database if exists DBProject;
create database DBProject;
show databases;
use DBProject;
show tables;
```

DATABASE AND TABLES

TABLE 1: STUDENTS

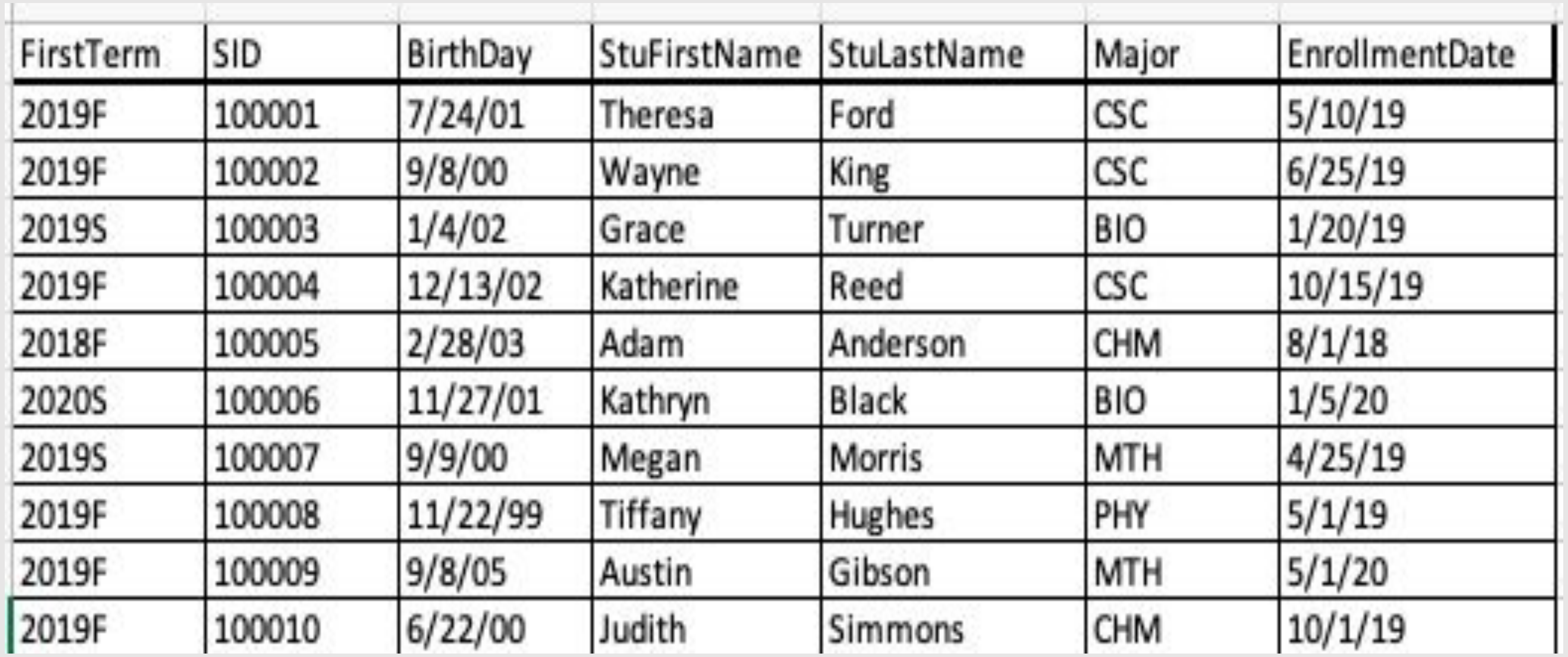
Contains records of every student that has been accepted, whether or not they actually enrolled in courses.



```sql
drop table if exists students;
create table students(
FirstTerm varchar(22) NOT NULL,
SID int PRIMARY KEY NOT NULL,
BirthDay date NOT NULL,
StuFirstName varchar(22) NOT NULL,
StuLastName varchar(22) NOT NULL,
major varchar(22) NOT NULL,
EnrollmentDate date NOT NULL);

insert into students values
('2019F', 100001 ,  '2001-07-24' ,   'Theresa',     'Ford',  'CSC' , '2019-05-10'),
('2019F', 100002 ,  '2000-09-08' ,     'Wayne',     'King',  'CSC' , '2019-06-25'),
('2019S', 100003 ,  '2002-01-04' ,     'Grace',   'Turner',  'BIO' , '2019-01-20'),
('2019F', 100004 ,  '2002-12-13' , 'Katherine',     'Reed',  'CSC' , '2019-10-15'),
('2018F', 100005 ,  '2003-02-28' ,      'Adam', 'Anderson',  'CHM' , '2018-08-01'),
('2020S', 100006 ,  '2001-11-27' ,   'Kathryn',    'Black',  'BIO' , '2020-01-05'),
('2019S', 100007 ,  '2000-09-09' ,     'Megan',   'Morris',  'MTH' , '2019-04-25'),
('2019F', 100008 ,  '1999-11-22' ,   'Tiffany',   'Hughes',  'PHY' , '2019-05-01'),
('2019F', 100009 ,  '2005-09-08' ,    'Austin',   'Gibson',  'MTH' , '2020-05-01'),
('2019F', 100010 ,  '2000-06-22' ,    'Judith',  'Simmons',  'CHM' , '2019-10-01');

select * from students;
```



FirstTerm	SID	BirthDay	StuFirstName	StuLastName	Major	EnrollmentDate
2019F	100001	7/24/01	Theresa	Ford	CSC	5/10/19
2019F	100002	9/8/00	Wayne	King	CSC	6/25/19
2019S	100003	1/4/02	Grace	Turner	BIO	1/20/19
2019F	100004	12/13/02	Katherine	Reed	CSC	10/15/19
2018F	100005	2/28/03	Adam	Anderson	CHM	8/1/18
2020S	100006	11/27/01	Kathryn	Black	BIO	1/5/20
2019S	100007	9/9/00	Megan	Morris	MTH	4/25/19
2019F	100008	11/22/99	Tiffany	Hughes	PHY	5/1/19
2019F	100009	9/8/05	Austin	Gibson	MTH	5/1/20
2019F	100010	6/22/00	Judith	Simmons	CHM	10/1/19

FirstTerm -
The term for which the student was accepted to enroll. This is recorded as a string containing the year followed by 'S' for Spring and 'F' for Fall

SID
Student ID

Birthday
The student's birthday

StuFirstName
The student's first name

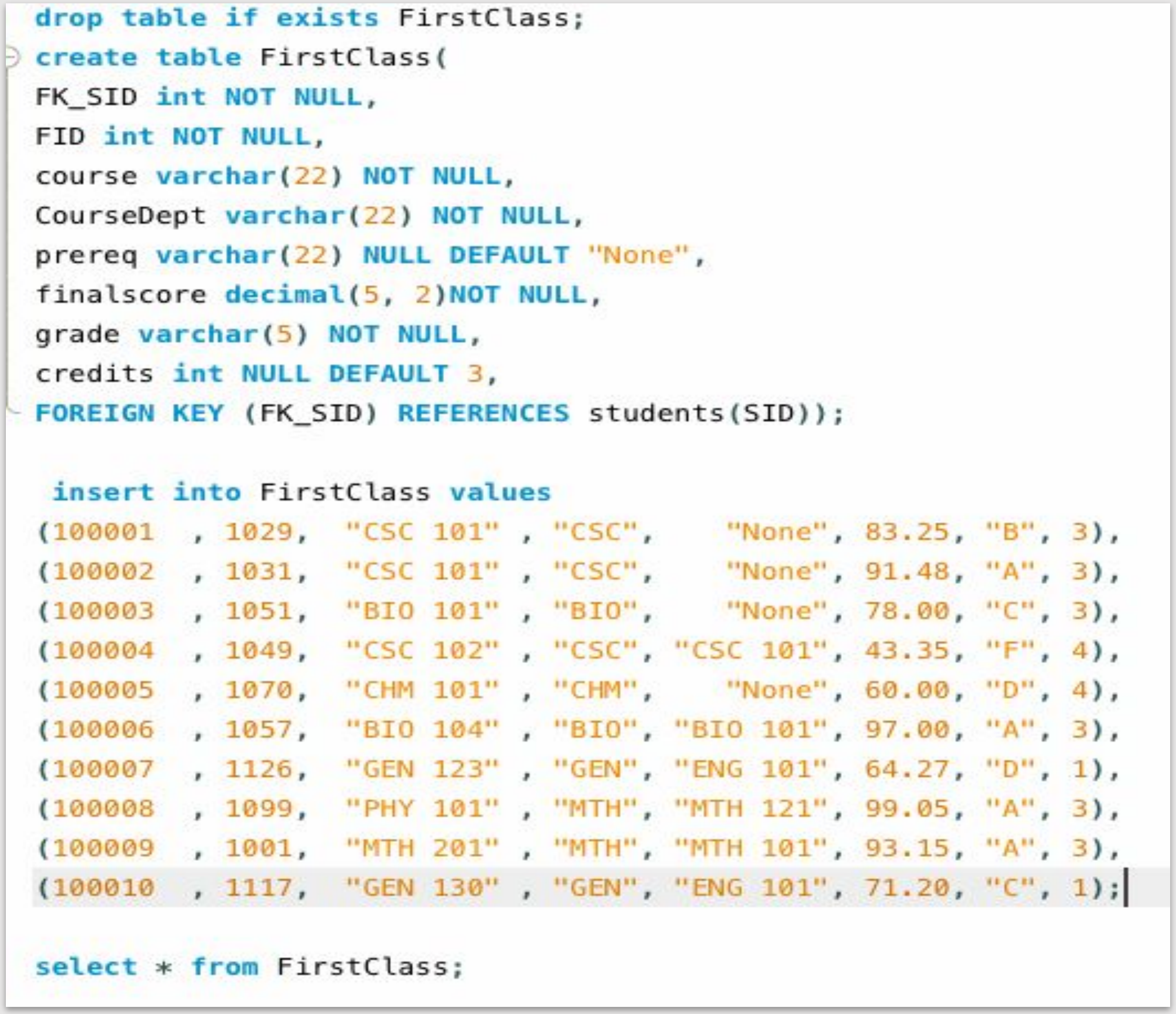
StuLastName
The student's last name

Major
A string representing the student's major. (BIO, CHM, CSC, MTH, or PHY)

Enrollment
Date student was enrolled at University

TABLE 2: FIRSTCLASS

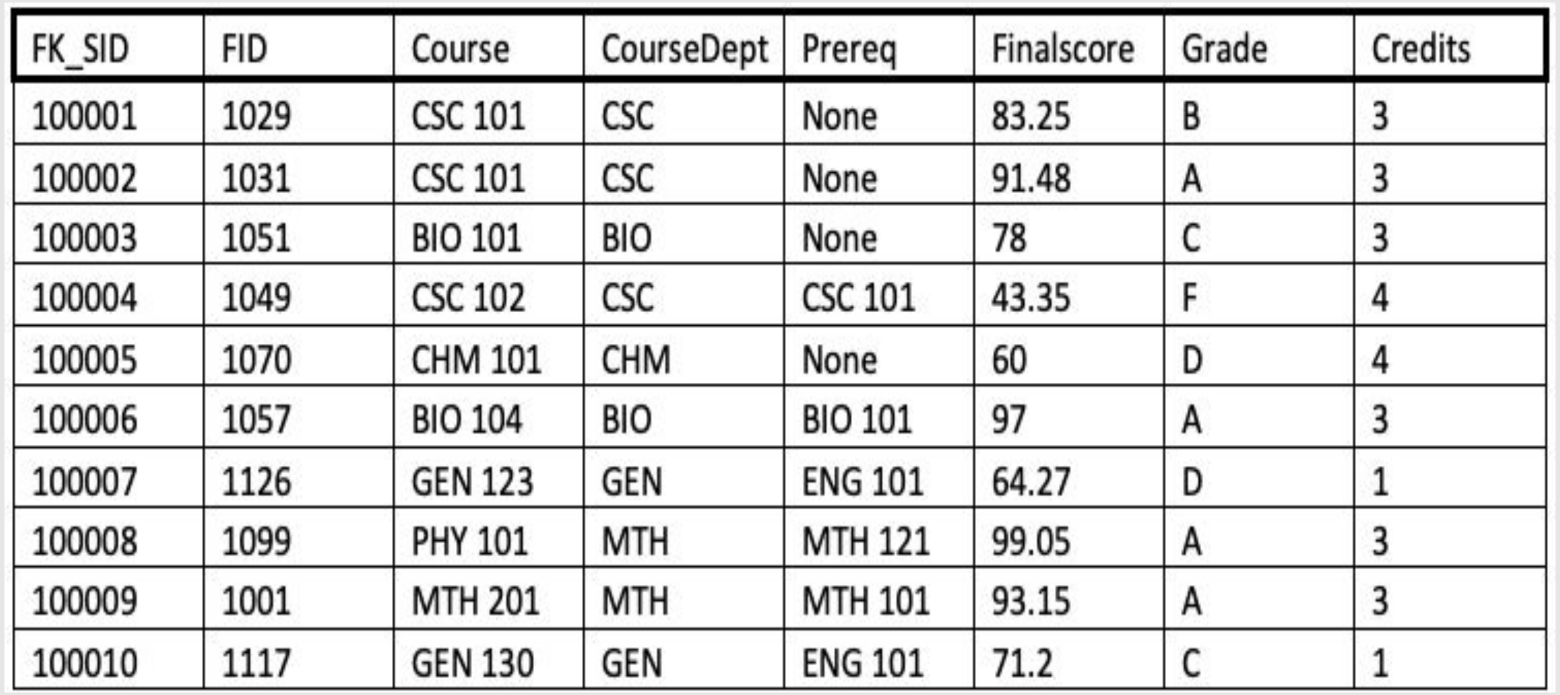
Class records -
Each row represents the information regarding the first class of a single student



```
drop table if exists FirstClass;
create table FirstClass(
FK_SID int NOT NULL,
FID int NOT NULL,
course varchar(22) NOT NULL,
CourseDept varchar(22) NOT NULL,
prereq varchar(22) NULL DEFAULT "None",
finalscore decimal(5, 2)NOT NULL,
grade varchar(5) NOT NULL,
credits int NULL DEFAULT 3,
FOREIGN KEY (FK_SID) REFERENCES students(SID));

 insert into FirstClass values
(100001  , 1029,  "CSC 101" , "CSC",    "None", 83.25, "B", 3),
(100002  , 1031,  "CSC 101" , "CSC",    "None", 91.48, "A", 3),
(100003  , 1051,  "BIO 101" , "BIO",    "None", 78.00, "C", 3),
(100004  , 1049,  "CSC 102" , "CSC", "CSC 101", 43.35, "F", 4),
(100005  , 1070,  "CHM 101" , "CHM",    "None", 60.00, "D", 4),
(100006  , 1057,  "BIO 104" , "BIO", "BIO 101", 97.00, "A", 3),
(100007  , 1126,  "GEN 123" , "GEN", "ENG 101", 64.27, "D", 1),
(100008  , 1099,  "PHY 101" , "MTH", "MTH 121", 99.05, "A", 3),
(100009  , 1001,  "MTH 201" , "MTH", "MTH 101", 93.15, "A", 3),
(100010  , 1117,  "GEN 130" , "GEN", "ENG 101", 71.20, "C", 1);

select * from FirstClass;
```



FK_SID	FID	Course	CourseDept	Prereq	Finalscore	Grade	Credits
100001	1029	CSC 101	CSC	None	83.25	B	3
100002	1031	CSC 101	CSC	None	91.48	A	3
100003	1051	BIO 101	BIO	None	78	C	3
100004	1049	CSC 102	CSC	CSC 101	43.35	F	4
100005	1070	CHM 101	CHM	None	60	D	4
100006	1057	BIO 104	BIO	BIO 101	97	A	3
100007	1126	GEN 123	GEN	ENG 101	64.27	D	1
100008	1099	PHY 101	MTH	MTH 121	99.05	A	3
100009	1001	MTH 201	MTH	MTH 101	93.15	A	3
100010	1117	GEN 130	GEN	ENG 101	71.2	C	1

FK_SID
The Student ID for the student earning the grade

FID
The Faculty ID for the course instructor

Course
The prefix and number of the course

CourseDept
The department that the course is offered in

Prereq
The prerequisite for the course (if any)

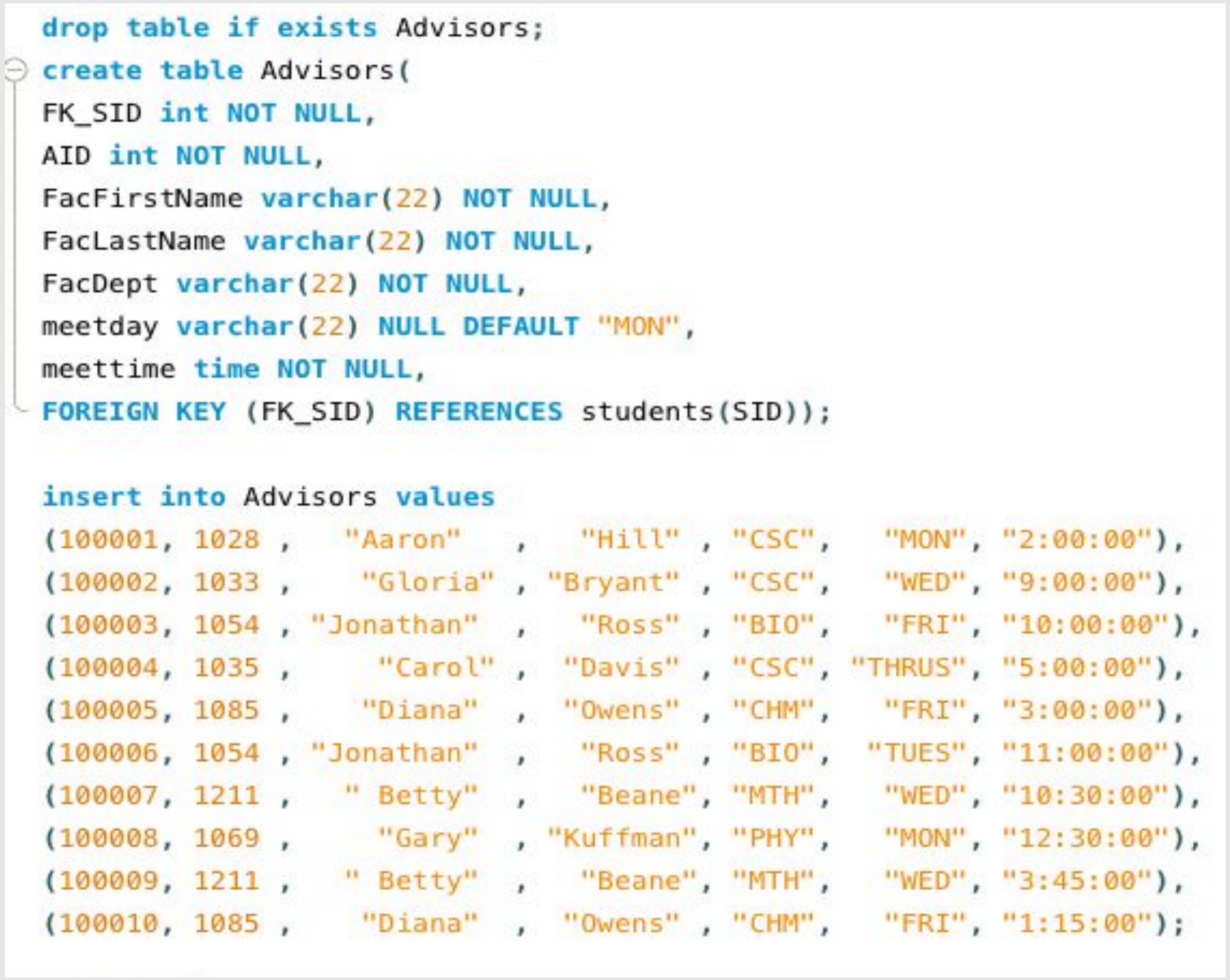
Final score
The number of points out of 100 that the student received as the final grade in the course

Grade
The letter grade that was given for the course

Credits
The number of credit hours for the course

TABLE 3: ADVISORS

Contains information regarding advisors for different students.



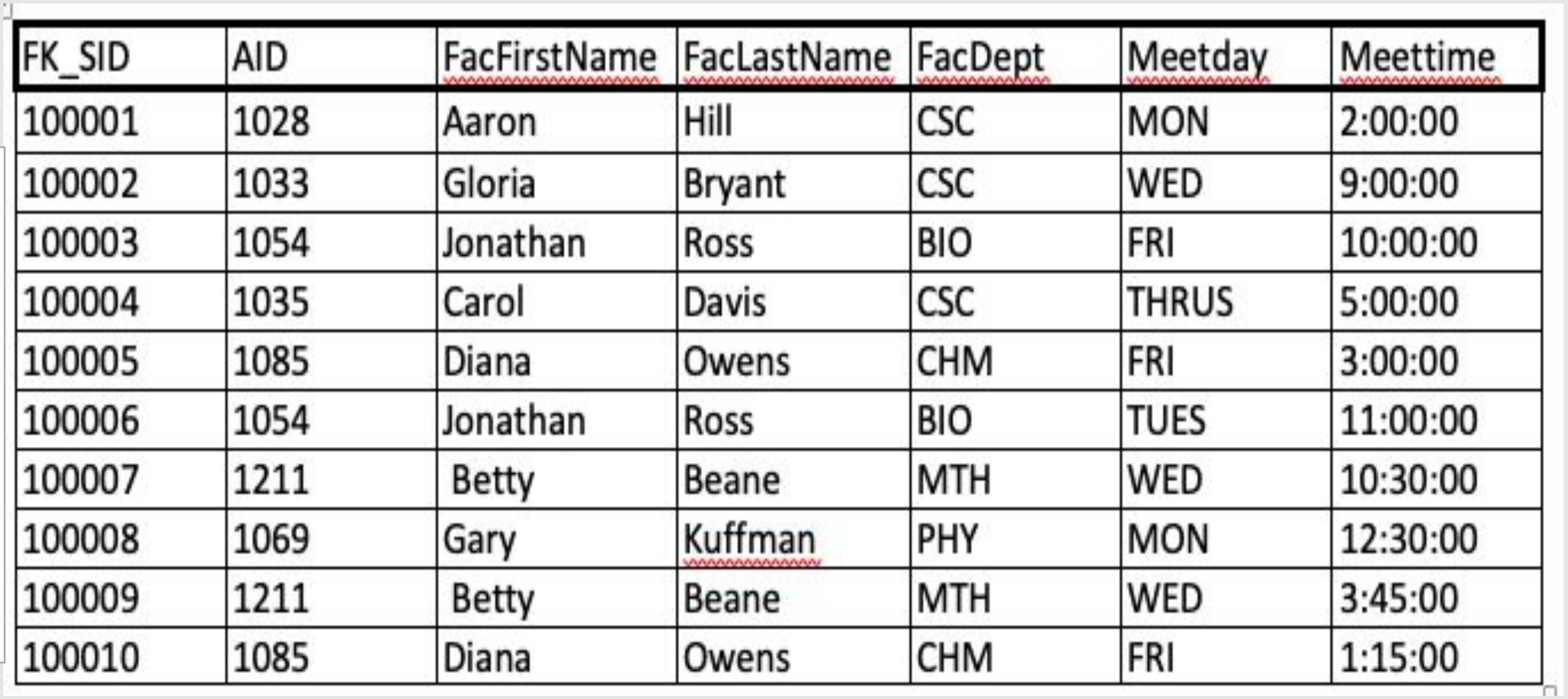
```
drop table if exists Advisors;
create table Advisors(
FK_SID int NOT NULL,
AID int NOT NULL,
FacFirstName varchar(22) NOT NULL,
FacLastName varchar(22) NOT NULL,
FacDept varchar(22) NOT NULL,
meetday varchar(22) NULL DEFAULT "MON",
meettime time NOT NULL,
FOREIGN KEY (FK_SID) REFERENCES students(SID));

insert into Advisors values
(100001, 1028 ,   "Aaron"    ,   "Hill" , "CSC",   "MON", "2:00:00"),
(100002, 1033 ,    "Gloria" , "Bryant" , "CSC",   "WED", "9:00:00"),
(100003, 1054 , "Jonathan"  ,   "Ross" , "BIO",   "FRI", "10:00:00"),
(100004, 1035 ,     "Carol" ,  "Davis" , "CSC", "THRUS", "5:00:00"),
(100005, 1085 ,    "Diana"  ,  "Owens" , "CHM",   "FRI", "3:00:00"),
(100006, 1054 , "Jonathan"  ,   "Ross" , "BIO",  "TUES", "11:00:00"),
(100007, 1211 ,   " Betty"  ,   "Beane", "MTH",   "WED", "10:30:00"),
(100008, 1069 ,     "Gary"  , "Kuffman", "PHY",   "MON", "12:30:00"),
(100009, 1211 ,   " Betty"  ,   "Beane", "MTH",   "WED", "3:45:00"),
(100010, 1085 ,    "Diana"  ,  "Owens" , "CHM",   "FRI", "1:15:00");
```

FK_SID
The SID for the student

AID
Advisor ID

FacFirstName
The faculty member's first name



FK_SID	AID	FacFirstName	FacLastName	FacDept	Meetday	Meettime
100001	1028	Aaron	Hill	CSC	MON	2:00:00
100002	1033	Gloria	Bryant	CSC	WED	9:00:00
100003	1054	Jonathan	Ross	BIO	FRI	10:00:00
100004	1035	Carol	Davis	CSC	THRUS	5:00:00
100005	1085	Diana	Owens	CHM	FRI	3:00:00
100006	1054	Jonathan	Ross	BIO	TUES	11:00:00
100007	1211	Betty	Beane	MTH	WED	10:30:00
100008	1069	Gary	Kuffman	PHY	MON	12:30:00
100009	1211	Betty	Beane	MTH	WED	3:45:00
100010	1085	Diana	Owens	CHM	FRI	1:15:00

FacLastName
The faculty member's last name

FacDept - The department the faculty member teaches in. (BIO, CHM, CSC, MTH, PHY, or GEN)

Meetday
The weekday on which the student and advisor meet

Meettime
The time of day at which the student and advisor meet

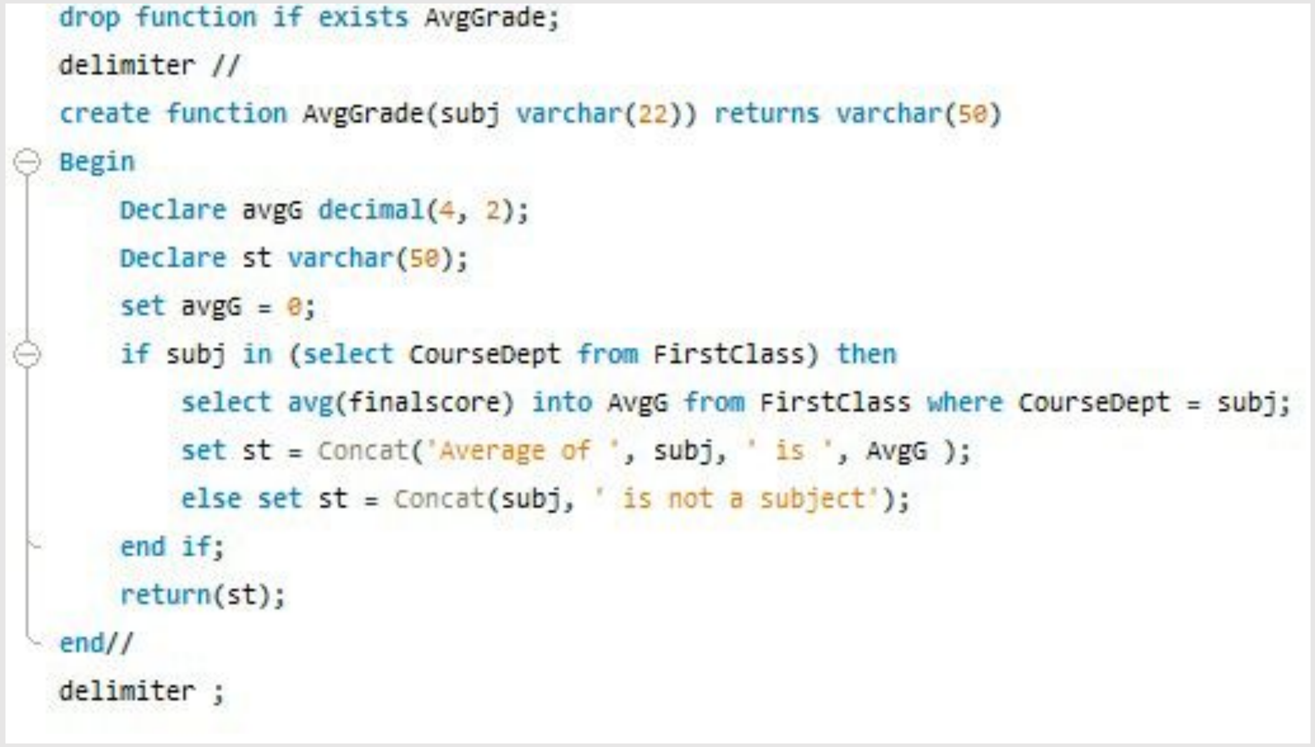
Preliminary Code

Before we code any procedures, functions, or triggers, we must use the following line of code:

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

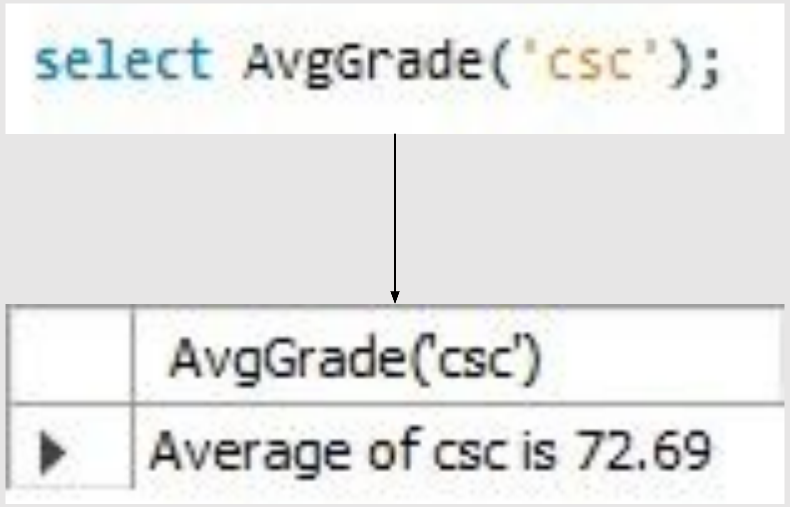
Function 1: Average Grade

This function will calculate the average grade for a given course:



```
drop function if exists AvgGrade;
delimiter //
create function AvgGrade(subj varchar(22)) returns varchar(50)
Begin
    Declare avgG decimal(4, 2);
    Declare st varchar(50);
    set avgG = 0;
    if subj in (select CourseDept from FirstClass) then
        select avg(finalscore) into AvgG from FirstClass where CourseDept = subj;
        set st = Concat('Average of ', subj, ' is ', AvgG );
        else set st = Concat(subj, ' is not a subject');
    end if;
    return(st);
end//
delimiter ;
```

Test Code:

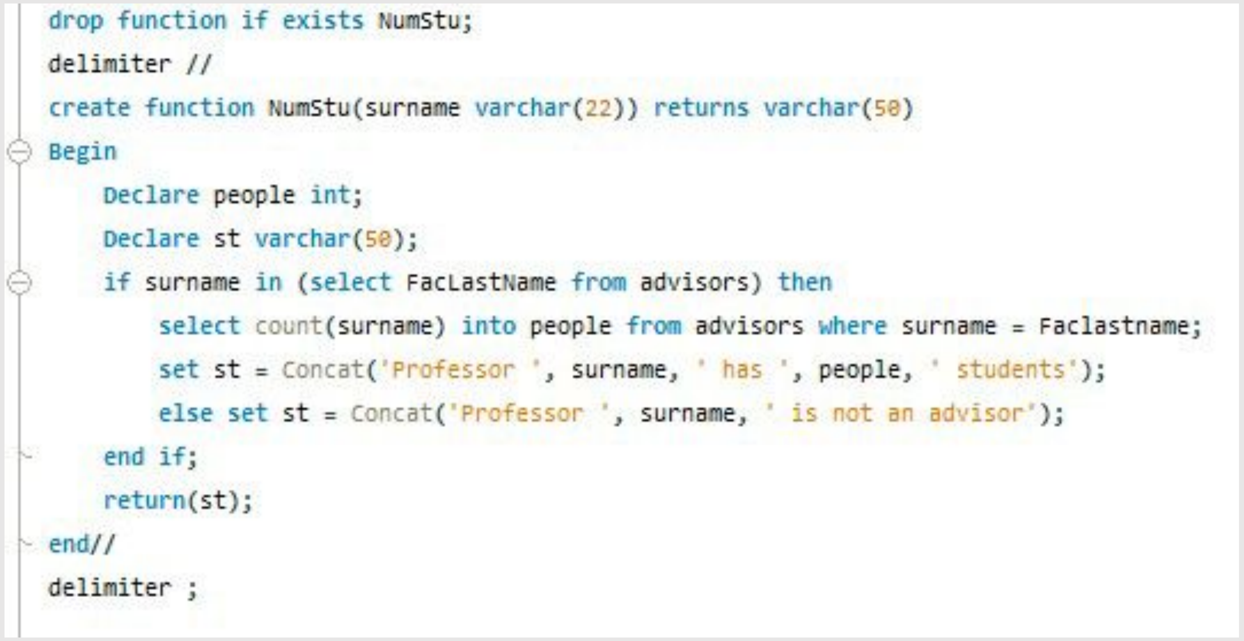


```
select AvgGrade('csc');
```

	AvgGrade('csc')
▶	Average of csc is 72.69

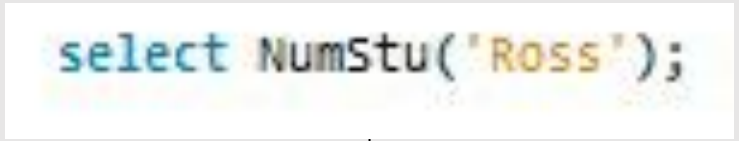
Function 2: Number of Students

This function will determine the number of students for a given advisor:

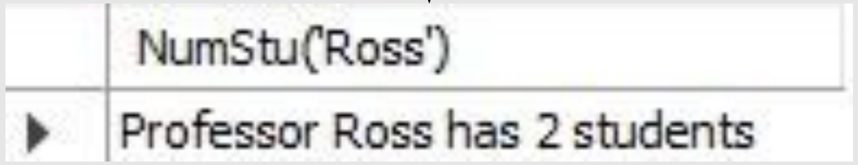


```
drop function if exists NumStu;
delimiter //
create function NumStu(surname varchar(22)) returns varchar(50)
Begin
    Declare people int;
    Declare st varchar(50);
    if surname in (select FacLastName from advisors) then
        select count(surname) into people from advisors where surname = Faclastname;
        set st = Concat('Professor ', surname, ' has ', people, ' students');
        else set st = Concat('Professor ', surname, ' is not an advisor');
    end if;
    return(st);
end//
delimiter ;
```

Test Code:



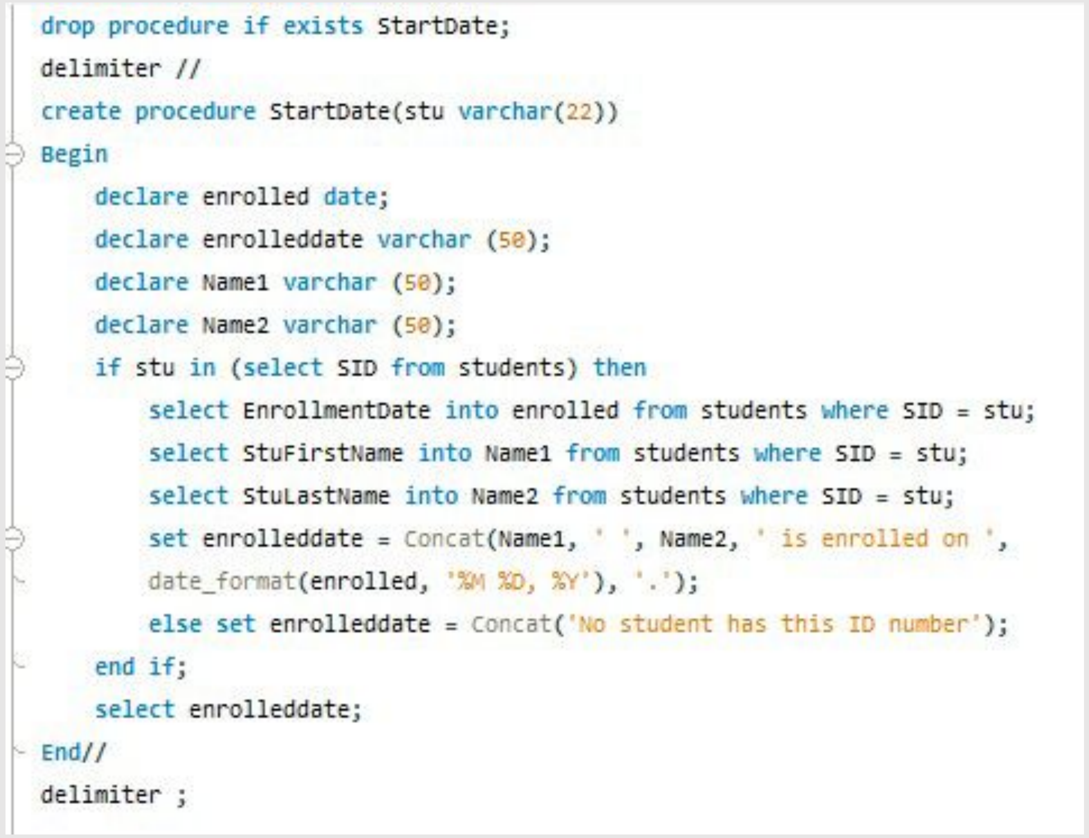
```
select NumStu('Ross');
```



```
   NumStu('Ross')
▶  Professor Ross has 2 students
```

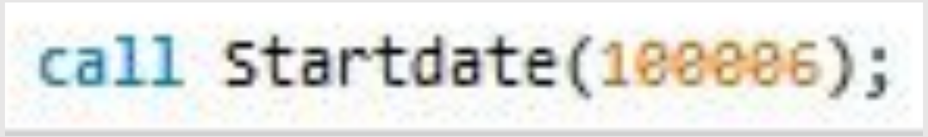
Procedure 1: Start Dates

This procedure will provide the date of enrollment for a given student ID:

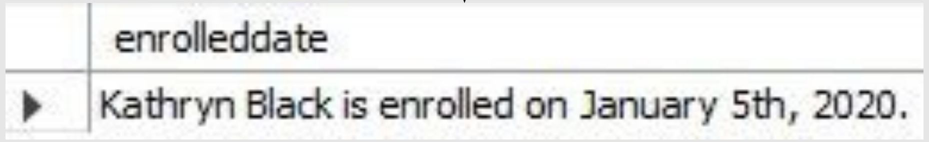


```
drop procedure if exists StartDate;
delimiter //
create procedure StartDate(stu varchar(22))
Begin
    declare enrolled date;
    declare enrolleddate varchar (50);
    declare Name1 varchar (50);
    declare Name2 varchar (50);
    if stu in (select SID from students) then
        select EnrollmentDate into enrolled from students where SID = stu;
        select StuFirstName into Name1 from students where SID = stu;
        select StuLastName into Name2 from students where SID = stu;
        set enrolleddate = Concat(Name1, ' ', Name2, ' is enrolled on ',
        date_format(enrolled, '%M %D, %Y'), '.');
        else set enrolleddate = Concat('No student has this ID number');
    end if;
    select enrolleddate;
End//
delimiter ;
```

Test Code:



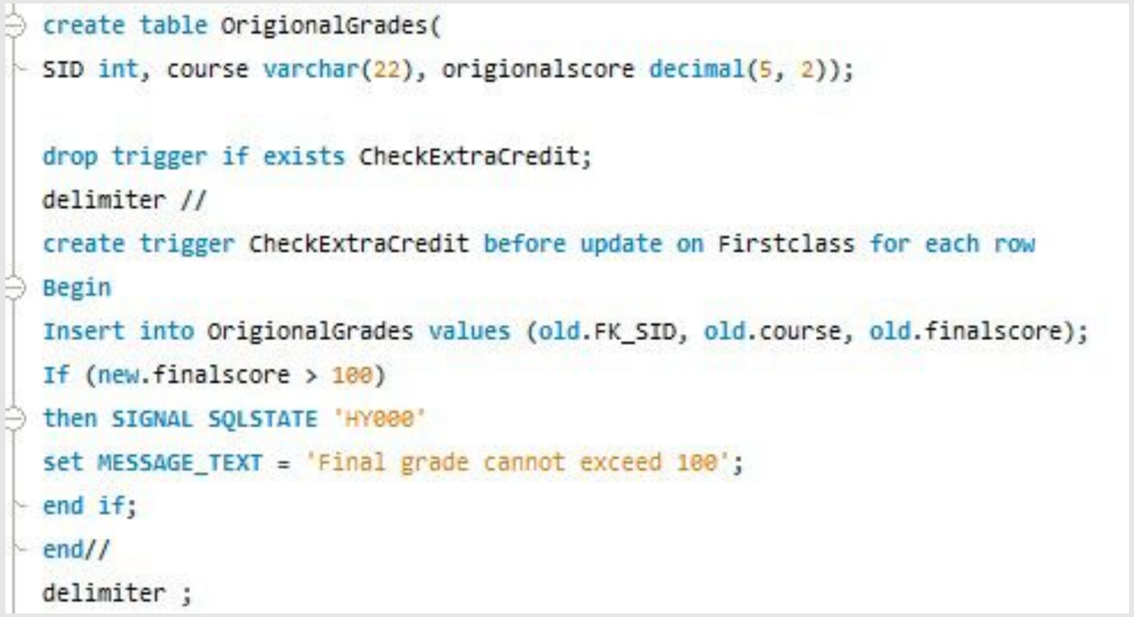
```
call Startdate(100006);
```



```
enrolleddate
Kathryn Black is enrolled on January 5th, 2020.
```

Trigger 1: Extra Credit Limit

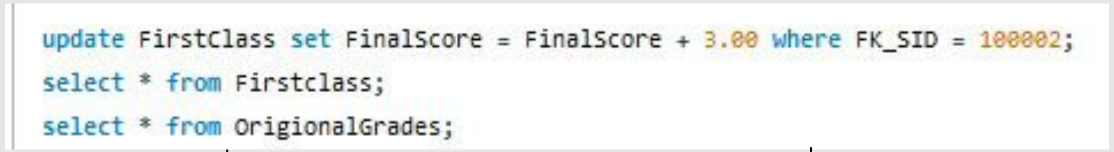
This trigger is for students who receive additional points on their final score from extra credit; it will ensure the final grade does not exceed 100 before updating, and it will store the original grades in a new table



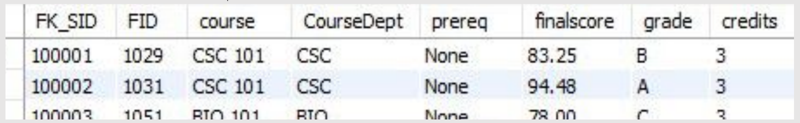
```
create table OrigionalGrades(
SID int, course varchar(22), origionalscore decimal(5, 2));

drop trigger if exists CheckExtraCredit;
delimiter //
create trigger CheckExtraCredit before update on Firstclass for each row
Begin
Insert into OrigionalGrades values (old.FK_SID, old.course, old.finalscore);
If (new.finalscore > 100)
then SIGNAL SQLSTATE 'HY000'
set MESSAGE_TEXT = 'Final grade cannot exceed 100';
end if;
end//
delimiter ;
```

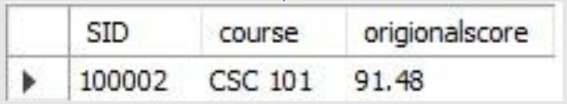
Valid Update (Original Grade of 91.48):



```
update FirstClass set FinalScore = FinalScore + 3.00 where FK_SID = 100002;
select * from Firstclass;
select * from OrigionalGrades;
```

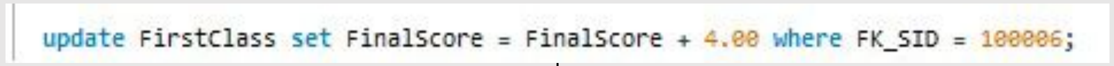


FK_SID	FID	course	CourseDept	prereq	finalscore	grade	credits
100001	1029	CSC 101	CSC	None	83.25	B	3
100002	1031	CSC 101	CSC	None	94.48	A	3
100003	1051	BIO 101	BIO	None	78.00	C	3

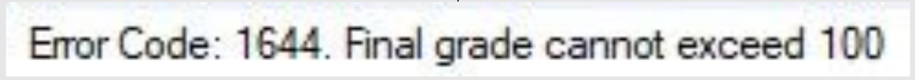


	SID	course	origionalscore
▶	100002	CSC 101	91.48

Invalid Update (Original Grade of 97.00):



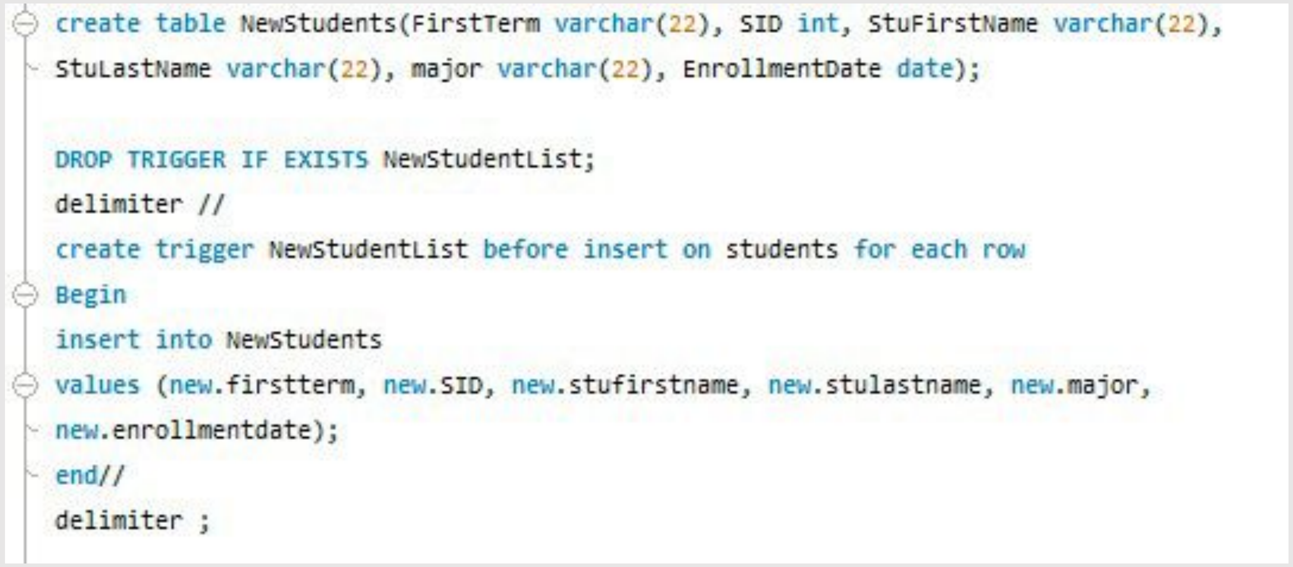
```
update FirstClass set FinalScore = FinalScore + 4.00 where FK_SID = 100006;
```



```
Error Code: 1644. Final grade cannot exceed 100
```

Trigger 2: New Students

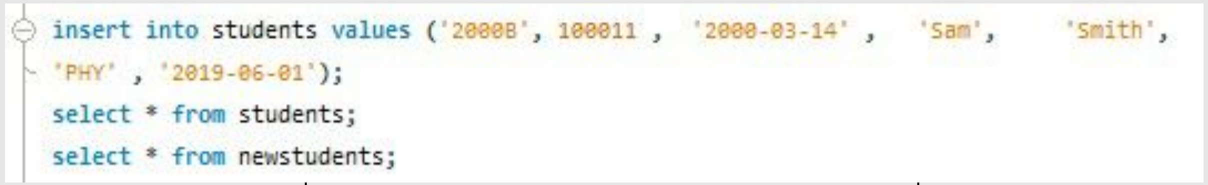
This trigger is for new students; it places their information into a new table before they are inserted into the Students table



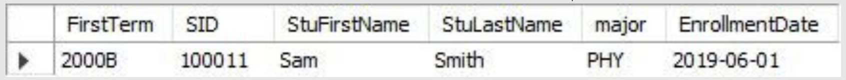
```
create table NewStudents(FirstTerm varchar(22), SID int, StuFirstName varchar(22),
StuLastName varchar(22), major varchar(22), EnrollmentDate date);

DROP TRIGGER IF EXISTS NewStudentList;
delimiter //
create trigger NewStudentList before insert on students for each row
Begin
insert into NewStudents
values (new.firstterm, new.SID, new.stufirstname, new.stulastname, new.major,
new.enrollmentdate);
end//
delimiter ;
```

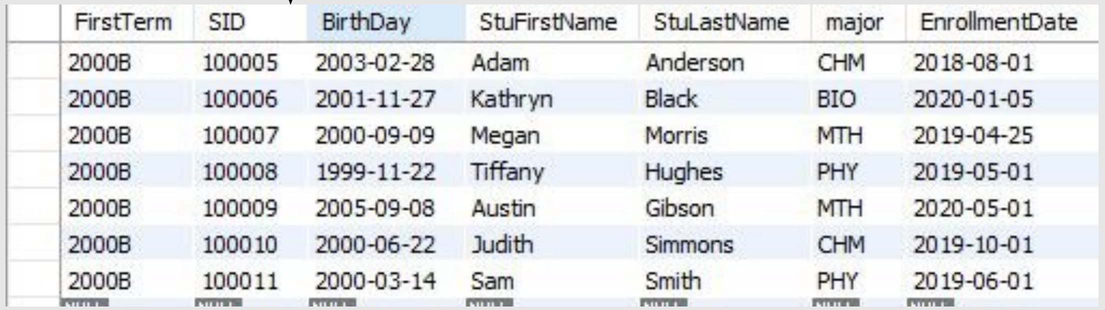
Test Code:



```
insert into students values ('2000B', 100011 ,  '2000-03-14' ,   'Sam',    'Smith',
'PHY' , '2019-06-01');
select * from students;
select * from newstudents;
```



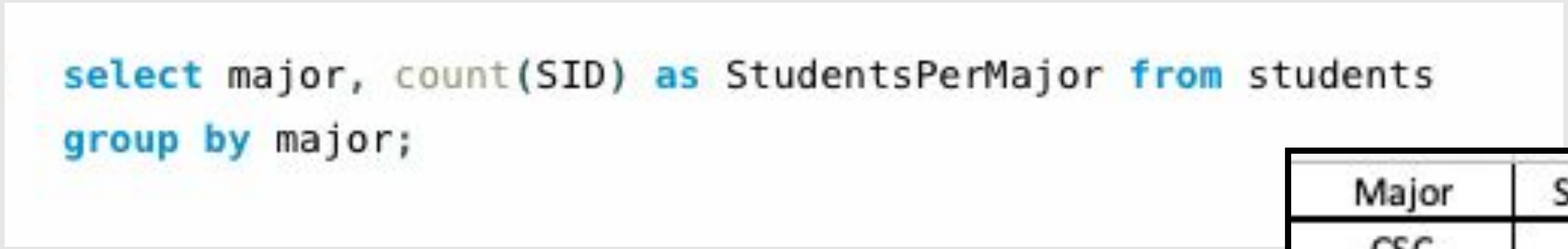
FirstTerm	SID	StuFirstName	StuLastName	major	EnrollmentDate
2000B	100011	Sam	Smith	PHY	2019-06-01



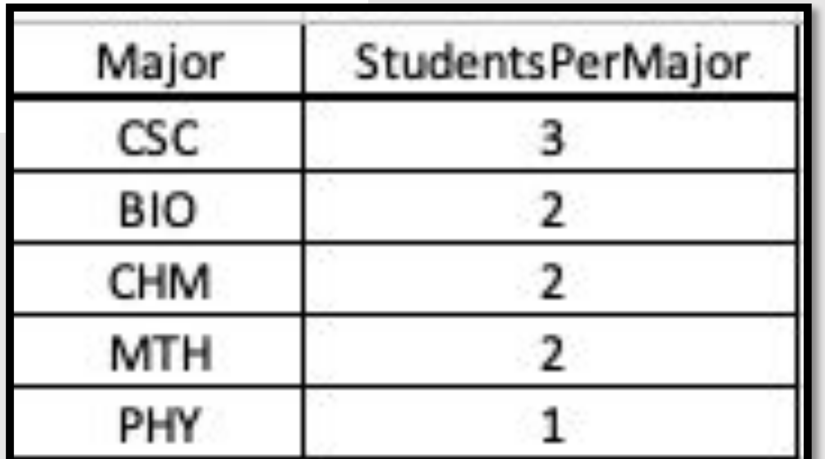
FirstTerm	SID	BirthDay	StuFirstName	StuLastName	major	EnrollmentDate
2000B	100005	2003-02-28	Adam	Anderson	CHM	2018-08-01
2000B	100006	2001-11-27	Kathryn	Black	BIO	2020-01-05
2000B	100007	2000-09-09	Megan	Morris	MTH	2019-04-25
2000B	100008	1999-11-22	Tiffany	Hughes	PHY	2019-05-01
2000B	100009	2005-09-08	Austin	Gibson	MTH	2020-05-01
2000B	100010	2000-06-22	Judith	Simmons	CHM	2019-10-01
2000B	100011	2000-03-14	Sam	Smith	PHY	2019-06-01

Summary Query 1: Total Students per Major

This summary query looks at the total number of students enrolled in each major in our database



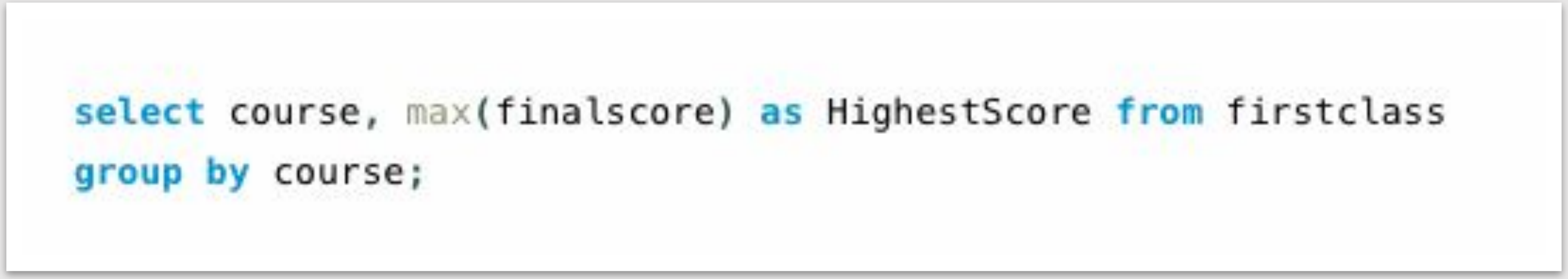
```
select major, count(SID) as StudentsPerMajor from students
group by major;
```



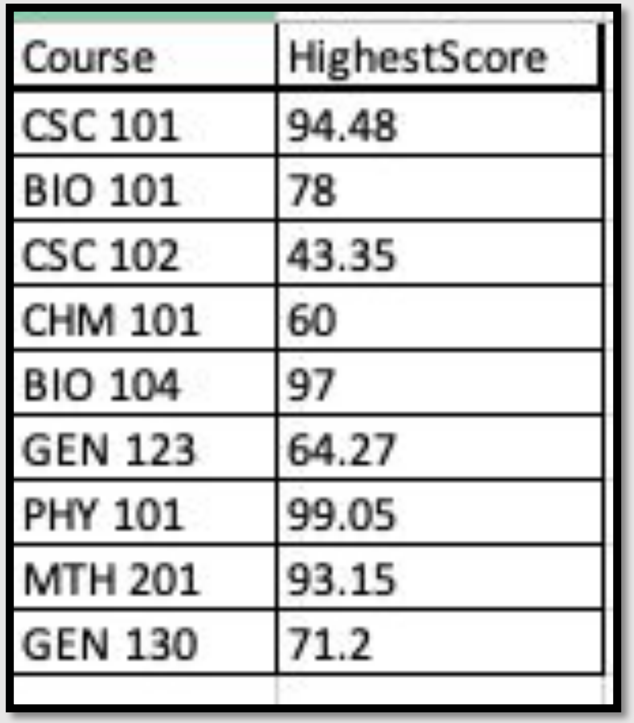
Major	StudentsPerMajor
CSC	3
BIO	2
CHM	2
MTH	2
PHY	1

Summary Query 2:Highest Score in Each Course

This summary query looks at the highest score obtained in each course



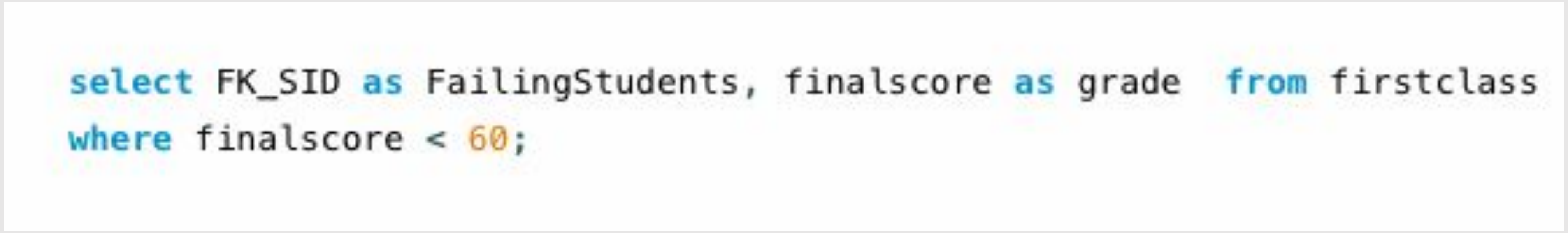
```sql
select course, max(finalscore) as HighestScore from firstclass
group by course;
```



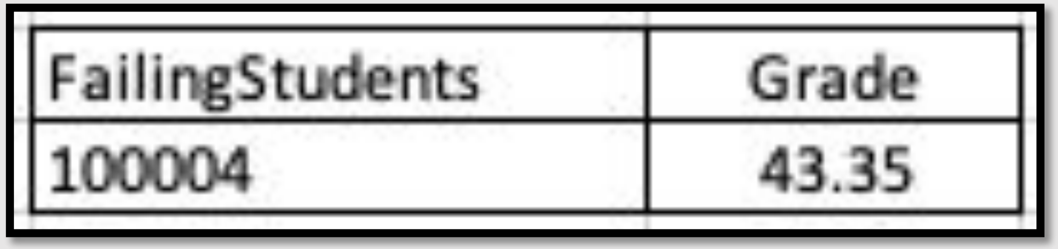
Course	HighestScore
CSC 101	94.48
BIO 101	78
CSC 102	43.35
CHM 101	60
BIO 104	97
GEN 123	64.27
PHY 101	99.05
MTH 201	93.15
GEN 130	71.2

Summary Query 3: Students Who Failed a Course (Final Score < 60)

This summary query looks at students with a failing grade (final score less than 60) in any the courses

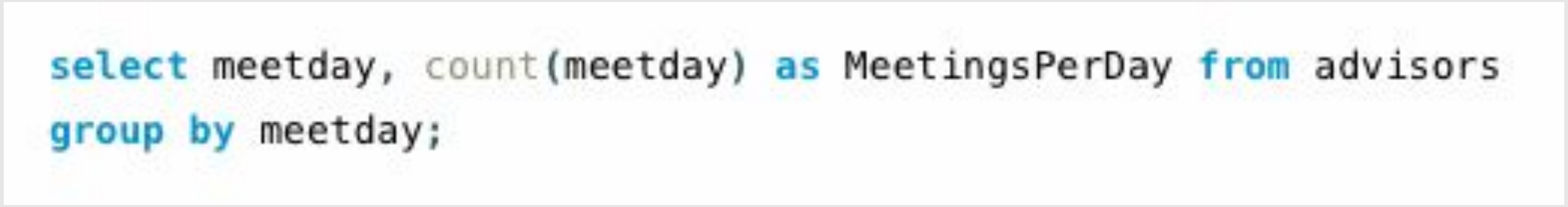


```
select FK_SID as FailingStudents, finalscore as grade  from firstclass
where finalscore < 60;
```



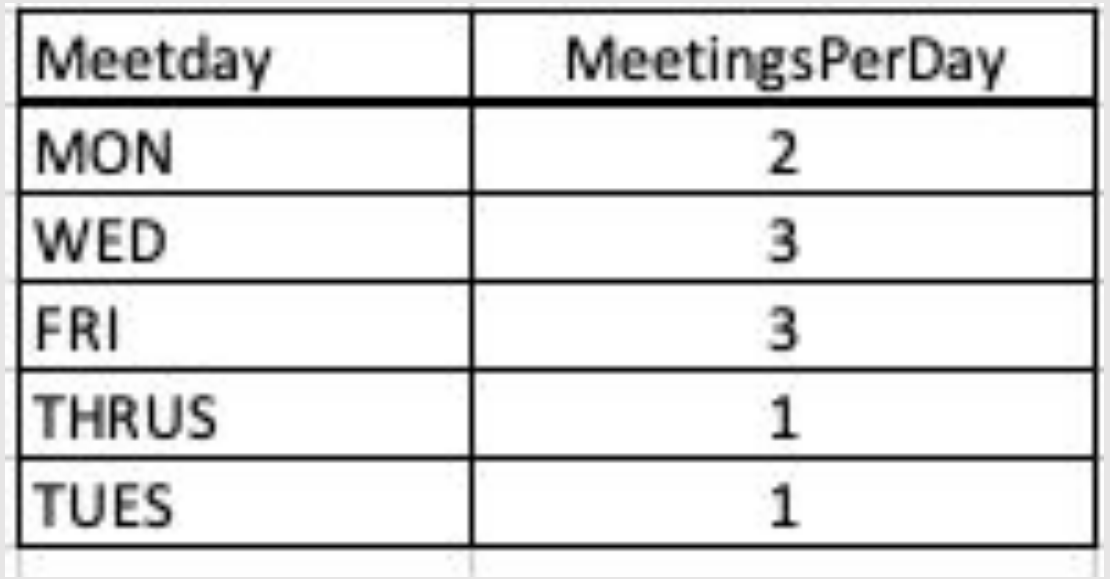
FailingStudents	Grade
100004	43.35

Summary Query 4 : Number of Advisor Meetings for Each Day of the Week



```
select meetday, count(meetday) as MeetingsPerDay from advisors
group by meetday;
```

This summary query looks at the number of student-advisor meetings for each day of the week



Meetday	MeetingsPerDay
MON	2
WED	3
FRI	3
THRUS	1
TUES	1

Conclusions

1. The functions and procedures we wrote let us easily retrieve any kind of information we want, like the average grade in different courses

2. We were able to implement all the concepts we have learned in this class in a real-life scenario, and our concepts would be even more useful on a larger set of data that a University would have access to

Thank you!

Any Questions?