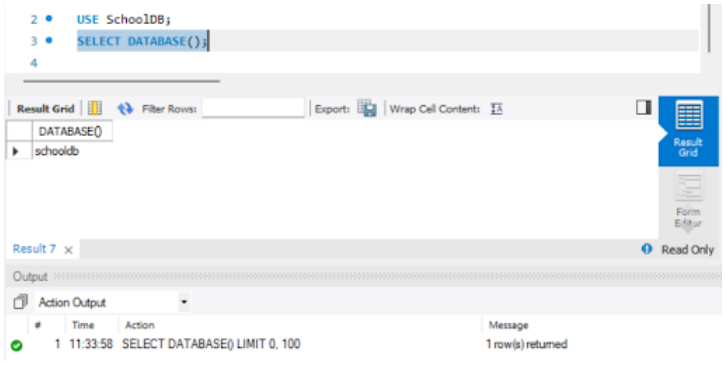
Ella El-Salahi
SQL Technical Document

Student Course Enrolment Database

Create Database

CREATE DATABASE *giving it a name* > USE *to make newly created database active* > SELECT DATABASE() *to check resulting active database in output*



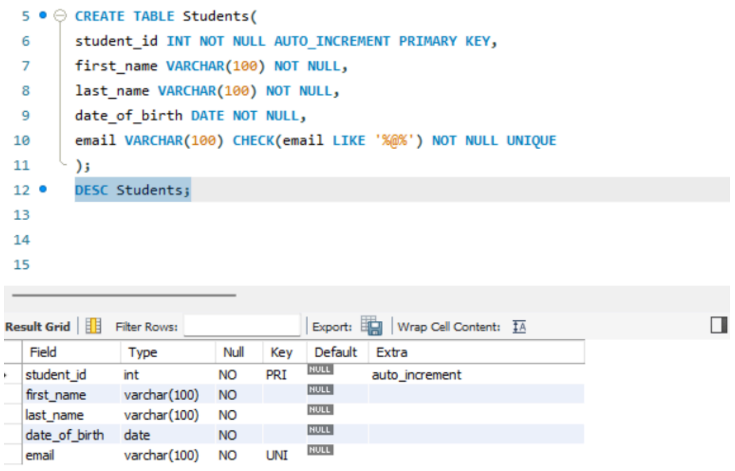
```
USE SchoolDB;
SELECT DATABASE();
```

DATABASE()
schooldb

```
#  Time      Action                          Message
1  11:33:58  SELECT DATABASE() LIMIT 0, 100  1 row(s) returned
```

Create / Populate Student Table

CREATE TABLE *giving it a name* > *Structure: adding fields and associated parameters FIELD_NAME | DATA TYPE | NULL/NOT NULL | KEY | DEFAULT | EXTRA* > PRIMARY KEY **required* assigned with a unique number that is populated automatically with* AUTO_INCREMENT > DESC/EXPLAIN Table *to show newly created table structure*

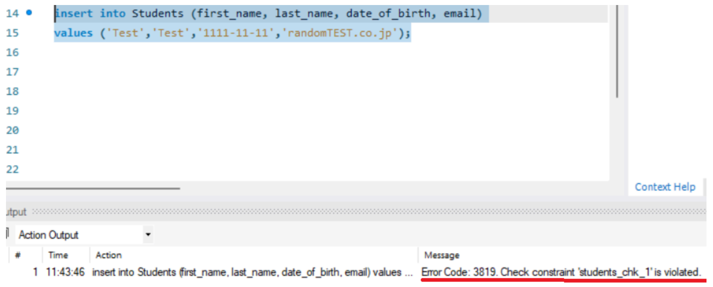


```
CREATE TABLE Students(
student_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(100) NOT NULL,
last_name VARCHAR(100) NOT NULL,
date_of_birth DATE NOT NULL,
email VARCHAR(100) CHECK(email LIKE '%@%') NOT NULL UNIQUE
);
DESC Students;
```

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	auto_increment
first_name	varchar(100)	NO		NULL	
last_name	varchar(100)	NO		NULL	
date_of_birth	date	NO		NULL	
email	varchar(100)	NO	UNI	NULL	

CONSTRAINT *from table creation* CHECK(email LIKE '%@%') *any values inserted into* email *field must have an '@' with an unknown amount of characters preceding and succeeding it - as indicated by the wildcard '%'*

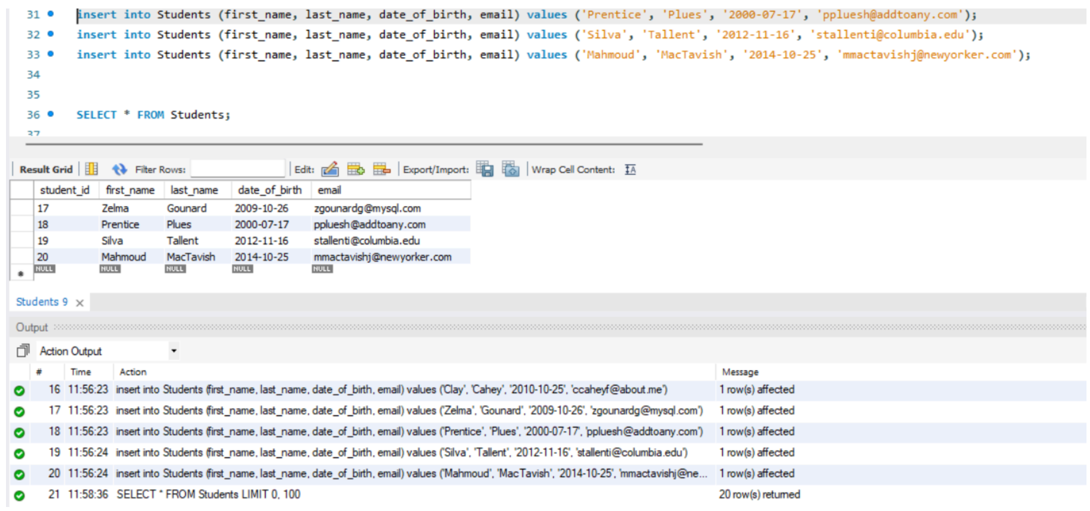
INSERT INTO *adding a record to the table without an '@' to check constraint is working >* Error Code Violation *shows constraint is working*



```sql
insert into Students (first_name, last_name, date_of_birth, email)
values ('Test','Test','1111-11-11','randomTEST.co.jp');
```

```
Action Output
#   Time      Action                                                              Message
1   11:43:46  insert into Students (first_name, last_name, date_of_birth, email) values ...   Error Code: 3819. Check constraint 'students_chk_1' is violated.
```

INSERT INTO *inserting data as values in associated fields > Syntax: Table_name (field_one, field_two, field_three..) values (field_one_value, field_two_value, field_three_value); >* SELECT * FROM *selecting all from the table to check all inserted correctly*



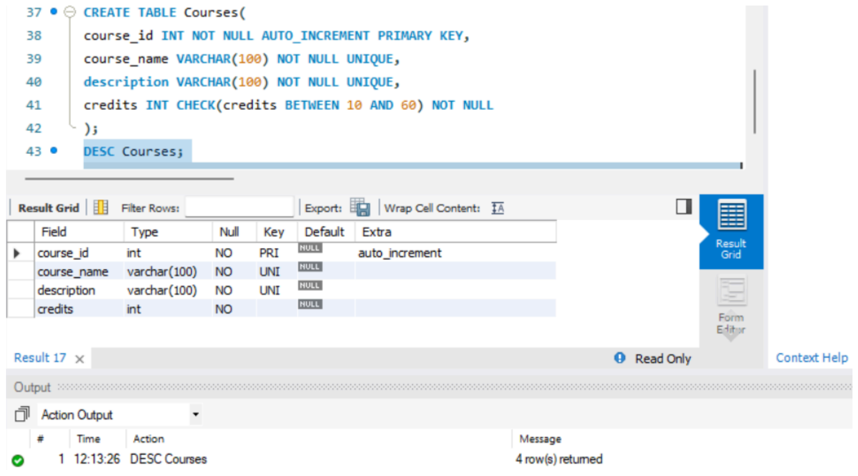
```sql
insert into Students (first_name, last_name, date_of_birth, email) values ('Prentice', 'Plues', '2000-07-17', 'ppluesh@addtoany.com');
insert into Students (first_name, last_name, date_of_birth, email) values ('Silva', 'Tallent', '2012-11-16', 'stallenti@columbia.edu');
insert into Students (first_name, last_name, date_of_birth, email) values ('Mahmoud', 'MacTavish', '2014-10-25', 'mmactavishj@newyorker.com');


SELECT * FROM Students;
```

student_id	first_name	last_name	date_of_birth	email
17	Zelma	Gounard	2009-10-26	zgounardg@mysql.com
18	Prentice	Plues	2000-07-17	ppluesh@addtoany.com
19	Silva	Tallent	2012-11-16	stallenti@columbia.edu
20	Mahmoud	MacTavish	2014-10-25	mmactavishj@newyorker.com
NULL	NULL	NULL	NULL	NULL

```
Action Output
#   Time      Action                                                                                                           Message
16  11:56:23  insert into Students (first_name, last_name, date_of_birth, email) values ('Clay', 'Cahey', '2010-10-25', 'ccaheyf@about.me')   1 row(s) affected
17  11:56:23  insert into Students (first_name, last_name, date_of_birth, email) values ('Zelma', 'Gounard', '2009-10-26', 'zgounardg@mysql.com')   1 row(s) affected
18  11:56:23  insert into Students (first_name, last_name, date_of_birth, email) values ('Prentice', 'Plues', '2000-07-17', 'ppluesh@addtoany.com')   1 row(s) affected
19  11:56:24  insert into Students (first_name, last_name, date_of_birth, email) values ('Silva', 'Tallent', '2012-11-16', 'stallenti@columbia.edu')   1 row(s) affected
20  11:56:24  insert into Students (first_name, last_name, date_of_birth, email) values ('Mahmoud', 'MacTavish', '2014-10-25', 'mmactavishj@ne...   1 row(s) affected
21  11:58:36  SELECT * FROM Students LIMIT 0, 100                                                                              20 row(s) returned
```

Create / Populate Course Table

CREATE TABLE > DESC/EXPLAIN *to check structure*

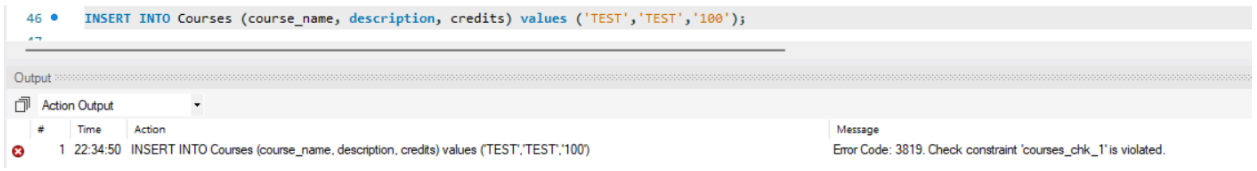


```sql
CREATE TABLE Courses(
course_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
course_name VARCHAR(100) NOT NULL UNIQUE,
description VARCHAR(100) NOT NULL UNIQUE,
credits INT CHECK(credits BETWEEN 10 AND 60) NOT NULL
);
DESC Courses;
```

Field	Type	Null	Key	Default	Extra
course_id	int	NO	PRI	NULL	auto_increment
course_name	varchar(100)	NO	UNI	NULL	
description	varchar(100)	NO	UNI	NULL	
credits	int	NO		NULL	

```
Action Output
#   Time      Action         Message
1   12:13:26  DESC Courses   4 row(s) returned
```

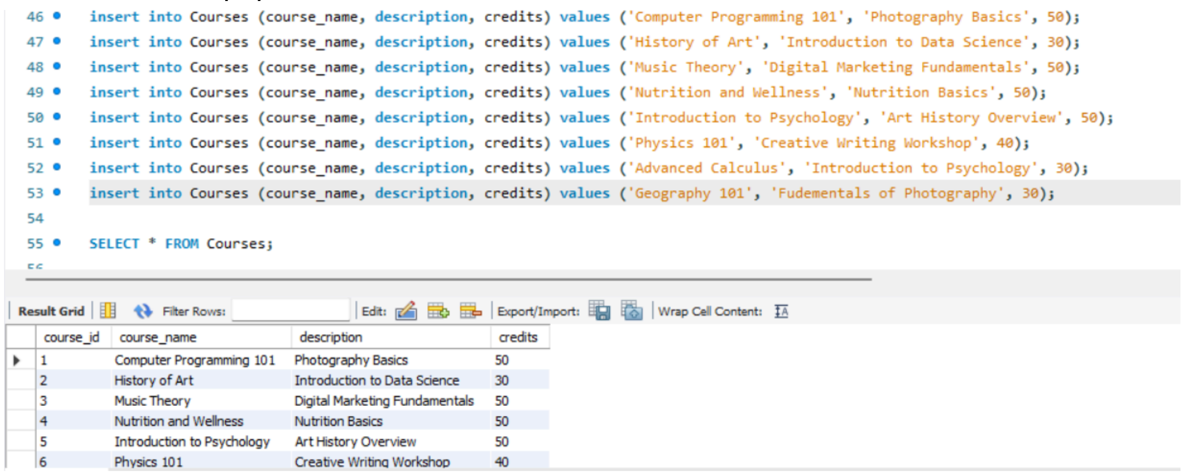
CHECK(credits BETWEEN 10 AND 60) *check constraint by attempting to insert* credit *value > 60*



```
INSERT INTO Courses (course_name, description, credits) values ('TEST','TEST','100');
```

```
Output
Action Output
#   Time      Action                                                                          Message
1   22:34:50  INSERT INTO Courses (course_name, description, credits) values ('TEST','TEST','100')   Error Code: 3819. Check constraint 'courses_chk_1' is violated.
```

INSERT INTO *to populate table* > SELECT * FROM to check table



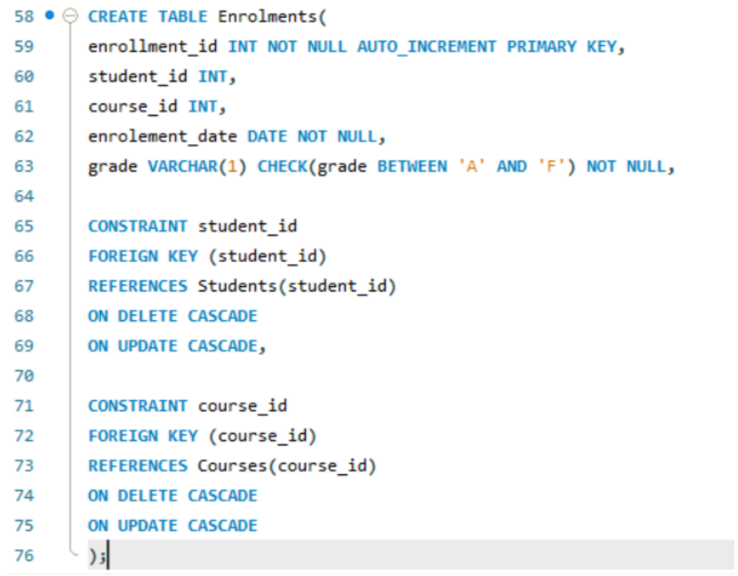
```
insert into Courses (course_name, description, credits) values ('Computer Programming 101', 'Photography Basics', 50);
insert into Courses (course_name, description, credits) values ('History of Art', 'Introduction to Data Science', 30);
insert into Courses (course_name, description, credits) values ('Music Theory', 'Digital Marketing Fundamentals', 50);
insert into Courses (course_name, description, credits) values ('Nutrition and Wellness', 'Nutrition Basics', 50);
insert into Courses (course_name, description, credits) values ('Introduction to Psychology', 'Art History Overview', 50);
insert into Courses (course_name, description, credits) values ('Physics 101', 'Creative Writing Workshop', 40);
insert into Courses (course_name, description, credits) values ('Advanced Calculus', 'Introduction to Psychology', 30);
insert into Courses (course_name, description, credits) values ('Geography 101', 'Fudementals of Photography', 30);

SELECT * FROM Courses;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

course_id	course_name	description	credits
1	Computer Programming 101	Photography Basics	50
2	History of Art	Introduction to Data Science	30
3	Music Theory	Digital Marketing Fundamentals	50
4	Nutrition and Wellness	Nutrition Basics	50
5	Introduction to Psychology	Art History Overview	50
6	Physics 101	Creative Writing Workshop	40

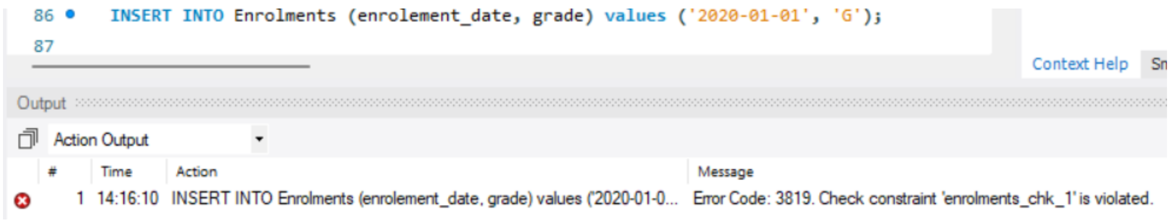
Create / Populate Enrolments Table

CREATE TABLE *Adding constraints to link* Students *tbl and* Courses *tbl to* Enrolments *tbl via foreign keys > Structure:*
CONSTRAINT constraint_name FOREIGN KEY field_name REFERENCES parent_table(field_name) > ON DELETE/UPDATE CASCADE *enacts delete/update in child table when action undertaken in the parent table*



```
CREATE TABLE Enrolments(
enrollment_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
student_id INT,
course_id INT,
enrolement_date DATE NOT NULL,
grade VARCHAR(1) CHECK(grade BETWEEN 'A' AND 'F') NOT NULL,

CONSTRAINT student_id
FOREIGN KEY (student_id)
REFERENCES Students(student_id)
ON DELETE CASCADE
ON UPDATE CASCADE,

CONSTRAINT course_id
FOREIGN KEY (course_id)
REFERENCES Courses(course_id)
ON DELETE CASCADE
ON UPDATE CASCADE
);
```

CHECK(grade BETWEEN ‘A’ AND ‘F’) *check constraint by attempting to insert* credit *value ‘G’*



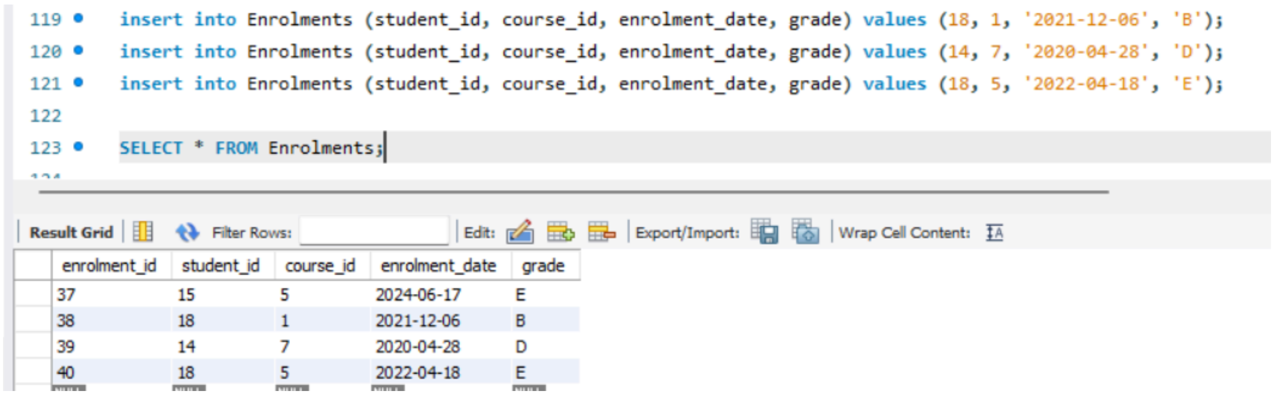
```
INSERT INTO Enrolments (enrolement_date, grade) values ('2020-01-01', 'G');
```

Output

Action Output

```
#  Time      Action                                                         Message
1  14:16:10  INSERT INTO Enrolments (enrolement_date, grade) values ('2020-01-0...  Error Code: 3819. Check constraint 'enrolments_chk_1' is violated.
```

INSERT INTO *to populate table >* student_id *field needs to be populated with values ranging between 1-20, as there are 20* student_id *values in* Students *table >* course_id field *needs to be populated with values ranging between 1-8 as there are 8* course_id *values in* Courses *table>* SELECT * FROM to check table

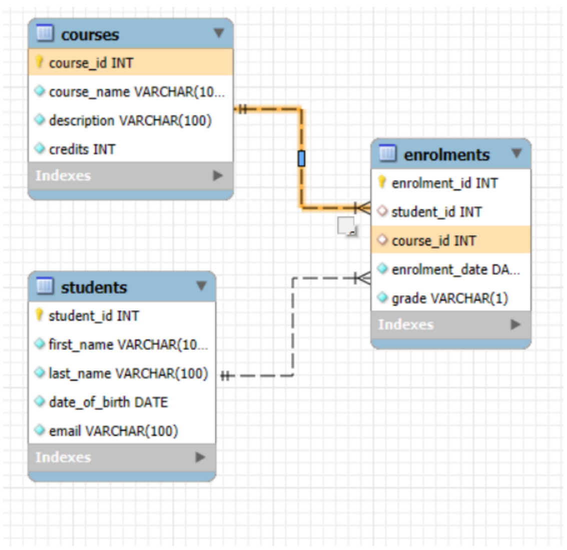


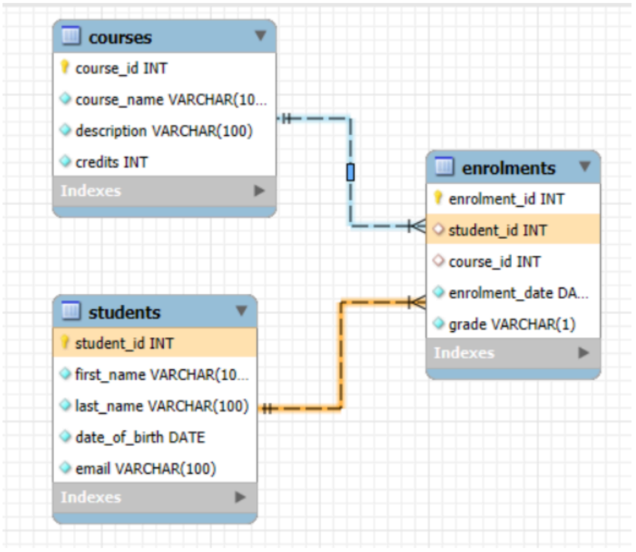
```
insert into Enrolments (student_id, course_id, enrolment_date, grade) values (18, 1, '2021-12-06', 'B');
insert into Enrolments (student_id, course_id, enrolment_date, grade) values (14, 7, '2020-04-28', 'D');
insert into Enrolments (student_id, course_id, enrolment_date, grade) values (18, 5, '2022-04-18', 'E');

SELECT * FROM Enrolments;
```

enrolment_id	student_id	course_id	enrolment_date	grade
37	15	5	2024-06-17	E
38	18	1	2021-12-06	B
39	14	7	2020-04-28	D
40	18	5	2022-04-18	E

Relational Tables

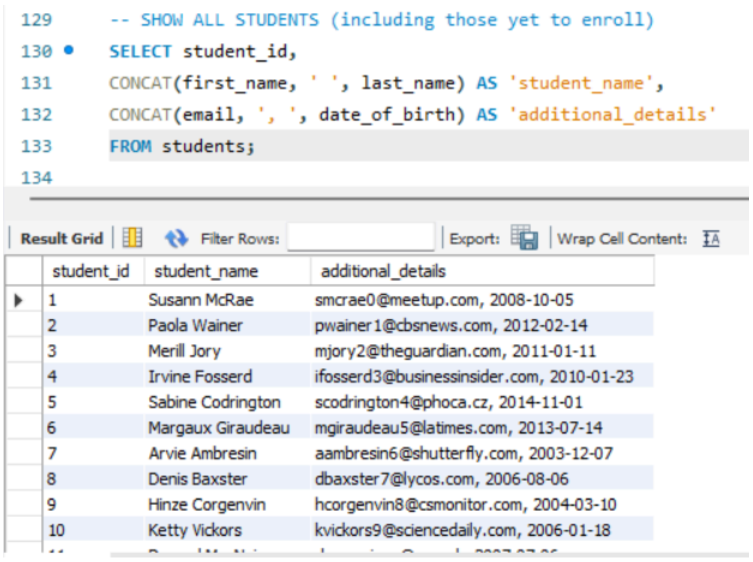
Parent_tbl . primary_key_field = child_tbl . foreign_key_field
(*parent*)Courses.course_id = (*child*)Enrolments.course_id
(*parent*)Students.student_id = (*child*)Enrolments.student_id





Query: All Students

Displaying all student's with associated student id's - incl student_is'd not present in Enrolment tbl (i.e. Students registered but yet to be enrolled)
SELECT _ FROM *selecting fields to display from* Students *table > CONCAT _ AS creating a temporary field by joining* 'first_name' *and* 'last_name' *and giving it an alias* 'student_name' > CONCAT _ AS *creating a temporary field by joining* 'email' *','* 'date_of_birth' *and giving it an alias* 'additional_details''



```
-- SHOW ALL STUDENTS (including those yet to enroll)
SELECT student_id,
CONCAT(first_name, ' ', last_name) AS 'student_name',
CONCAT(email, ', ', date_of_birth) AS 'additional_details'
FROM students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

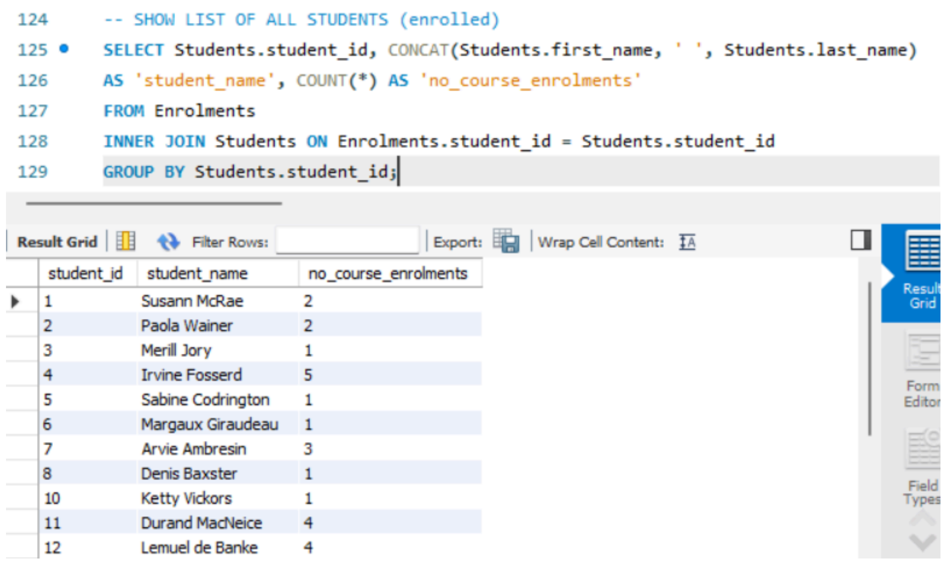
student_id	student_name	additional_details
1	Susann McRae	smcrae0@meetup.com, 2008-10-05
2	Paola Wainer	pwainer1@cbsnews.com, 2012-02-14
3	Merill Jory	mjory2@theguardian.com, 2011-01-11
4	Irvine Fosserd	ifosserd3@businessinsider.com, 2010-01-23
5	Sabine Codrington	scodrington4@phoca.cz, 2014-11-01
6	Margaux Giraudeau	mgiraudeau5@latimes.com, 2013-07-14
7	Arvie Ambresin	aambresin6@shutterfly.com, 2003-12-07
8	Denis Baxster	dbaxster7@lycos.com, 2006-08-06
9	Hinze Corgenvin	hcorgenvin8@csmonitor.com, 2004-03-10
10	Ketty Vickors	kvickors9@sciencedaily.com, 2006-01-18

CONCAT _ AS Syntax:

```
CONCAT(field_name1, ' ', field_name2) AS 'new_field_name'
```

Query: All Enrolled Students

Displaying all student's with associated student ids' and no. of course they are enrolled on, from Enrolment *tbl*
SELECT _ FROM *selecting fields to display from* Students *tbl and* Enrolments *tbl* > CONCAT _ AS *creating temporary field from* first_name *and* last_name *as* 'student_name' > COUNT(*) _ AS_GROUP BY *Creating a new field that counts all records and groups them by* student_id *(how many records in* Enrolment *per student). This number per student denotes how many course enrolments each student has had* > INNER JOIN *Selecting records from* Students *tbl via matching values in foreign key(child) = primary key(parent)*



```sql
-- SHOW LIST OF ALL STUDENTS (enrolled)
SELECT Students.student_id, CONCAT(Students.first_name, ' ', Students.last_name)
AS 'student_name', COUNT(*) AS 'no_course_enrolments'
FROM Enrolments
INNER JOIN Students ON Enrolments.student_id = Students.student_id
GROUP BY Students.student_id;
```

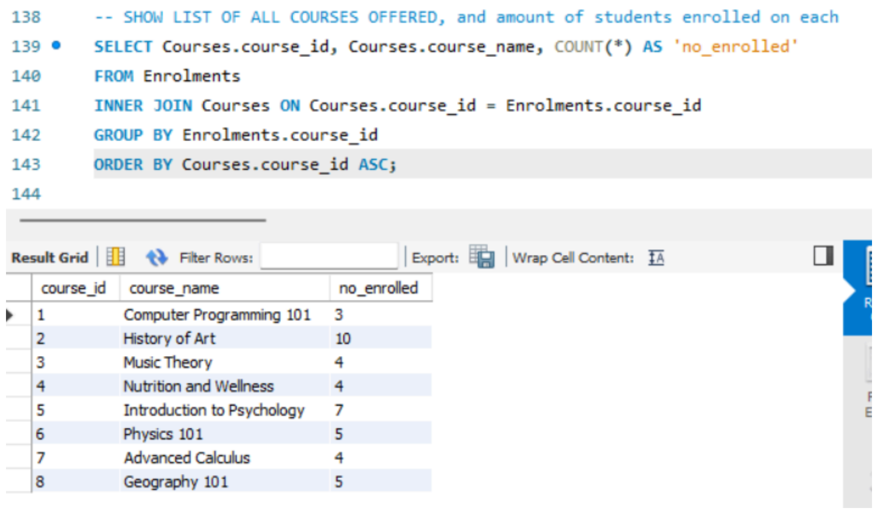
student_id	student_name	no_course_enrolments
1	Susann McRae	2
2	Paola Wainer	2
3	Merill Jory	1
4	Irvine Fosserd	5
5	Sabine Codrington	1
6	Margaux Giraudeau	1
7	Arvie Ambresin	3
8	Denis Baxster	1
10	Ketty Vickors	1
11	Durand MacNeice	4
12	Lemuel de Banke	4

INNER JOIN Syntax:

```sql
INNER JOIN primary_table ON secondary_table.foreign_key = primary_table.primary_key
```

Query: All Courses

Displaying all courses offered. with associated course ids' and no. of students enrolled, from Enrolment *tbl*
SELECT _ FROM *selecting fields to display from* Courses *tbl and* Enrolments *tbl* > COUNT(*) _AS_GROUP BY *creating a field that counts all records in* Enrolments *tbl and groups them by* course_id *(how many records in* Enrolment *per course). This number per course denotes how many students enrolments each course has had* > ORDER BY_ASC *Orders the table by* course_id *number in ascending order*



```sql
-- SHOW LIST OF ALL COURSES OFFERED, and amount of students enrolled on each
SELECT Courses.course_id, Courses.course_name, COUNT(*) AS 'no_enrolled'
FROM Enrolments
INNER JOIN Courses ON Courses.course_id = Enrolments.course_id
GROUP BY Enrolments.course_id
ORDER BY Courses.course_id ASC;
```

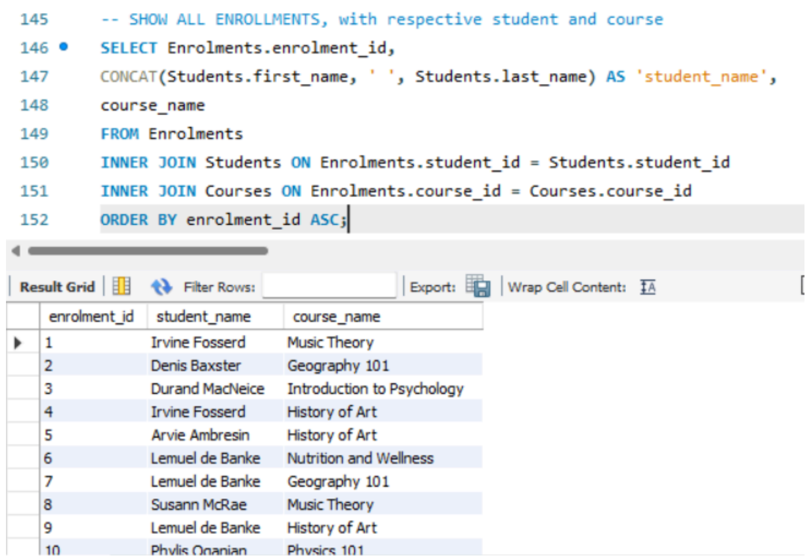
course_id	course_name	no_enrolled
1	Computer Programming 101	3
2	History of Art	10
3	Music Theory	4
4	Nutrition and Wellness	4
5	Introduction to Psychology	7
6	Physics 101	5
7	Advanced Calculus	4
8	Geography 101	5

ORDER BY _ Syntax:

```sql
-- ORDER BY field_name ASC/DESC
```

Query: All Enrollments

Displaying all enrolments, with respective student and course
SELECT _ FROM *selecting fields to display from* Students *tbl,* Courses *tbl and* Enrolments *tbl* > CONCAT_AS *to create temp fields* 'student_name' > INNER JOIN *Two inner joins to access both* Students *tbl and* Courses *tbl* records > ORDER BY _ *order by* enrolemnt_id *in ascending numerical order*



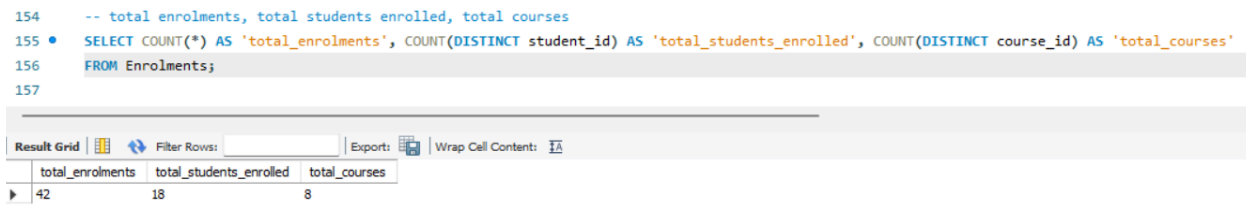
```sql
-- SHOW ALL ENROLLMENTS, with respective student and course
SELECT Enrolments.enrolment_id,
CONCAT(Students.first_name, ' ', Students.last_name) AS 'student_name',
course_name
FROM Enrolments
INNER JOIN Students ON Enrolments.student_id = Students.student_id
INNER JOIN Courses ON Enrolments.course_id = Courses.course_id
ORDER BY enrolment_id ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

enrolment_id	student_name	course_name
1	Irvine Fosserd	Music Theory
2	Denis Baxster	Geography 101
3	Durand MacNeice	Introduction to Psychology
4	Irvine Fosserd	History of Art
5	Arvie Ambresin	History of Art
6	Lemuel de Banke	Nutrition and Wellness
7	Lemuel de Banke	Geography 101
8	Susann McRae	Music Theory
9	Lemuel de Banke	History of Art
10	Phylis Oganian	Physics 101

Query: Total Enrollments

Displaying the total amount of enrollments, with the number of students enrolled on courses and the total number of course
COUNT(*)_AS *creating a field that counts all records in* Enrolments > COUNT(DISTINCT _) *creating a field that counts all records, but eliminates duplicates of specified field*



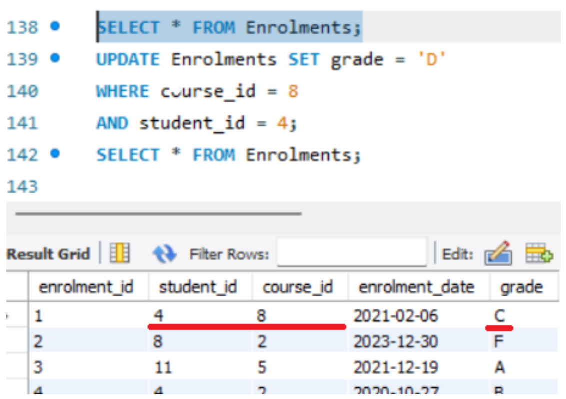
```sql
-- total enrolments, total students enrolled, total courses
SELECT COUNT(*) AS 'total_enrolments', COUNT(DISTINCT student_id) AS 'total_students_enrolled', COUNT(DISTINCT course_id) AS 'total_courses'
FROM Enrolments;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_enrolments	total_students_enrolled	total_courses
42	18	8

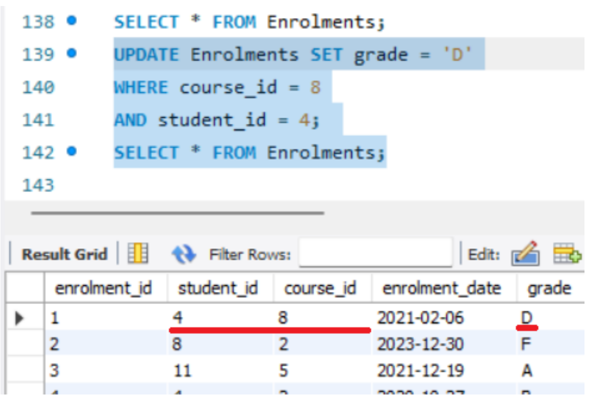
Update a Student's Grade

Manually update considering you know the students' student_id, course_id
SELECT _ FROM *functioning to give before/after comparison* > UPDATE _ SET_WHERE_AND *Updating data from* Enrolments *tbl, setting the new data to 'D' where conditions are both met - where* course_id = *8 and where* student_id = *4*



```sql
SELECT * FROM Enrolments;
UPDATE Enrolments SET grade = 'D'
WHERE course_id = 8
AND student_id = 4;
SELECT * FROM Enrolments;
```

enrolment_id	student_id	course_id	enrolment_date	grade
1	4	8	2021-02-06	C
2	8	2	2023-12-30	F
3	11	5	2021-12-19	A

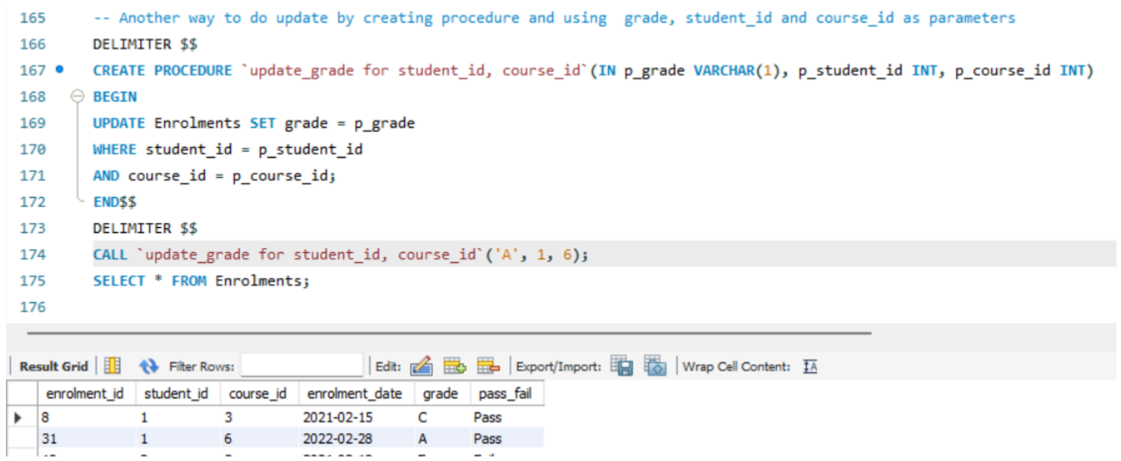


```sql
SELECT * FROM Enrolments;
UPDATE Enrolments SET grade = 'D'
WHERE course_id = 8
AND student_id = 4;
SELECT * FROM Enrolments;
```

enrolment_id	student_id	course_id	enrolment_date	grade
1	4	8	2021-02-06	D
2	8	2	2023-12-30	F
3	11	5	2021-12-19	A

Update a Student's Grade - **Stored Procedure**

Placing block of code to update student's grade into a stored procedure so it can be repeated as many times as needed, using grade, student_id *and* course_id *as parameters to be inputted* DELIMITER $$ *redefining delimiter to char $$ to avoid syntax error/incorrect execution of compound code within procedure* > CREATE PROCEDURE *giving procedure name, and specifying parameters* grade, student_id, course_id *and their data types* > BEGIN _ END *with code to be executed in-between* > UPDATE _SET_WHERE_AND *Code to be executed, substituting parameters for values entered when stored procedure called* > CALL *enacting the stored procedure, with values for* grade, student_id and course_id



```sql
-- Another way to do update by creating procedure and using  grade, student_id and course_id as parameters
DELIMITER $$
CREATE PROCEDURE `update_grade for student_id, course_id`(IN p_grade VARCHAR(1), p_student_id INT, p_course_id INT)
BEGIN
UPDATE Enrolments SET grade = p_grade
WHERE student_id = p_student_id
AND course_id = p_course_id;
END$$
DELIMITER $$
CALL `update_grade for student_id, course_id`('A', 1, 6);
SELECT * FROM Enrolments;
```

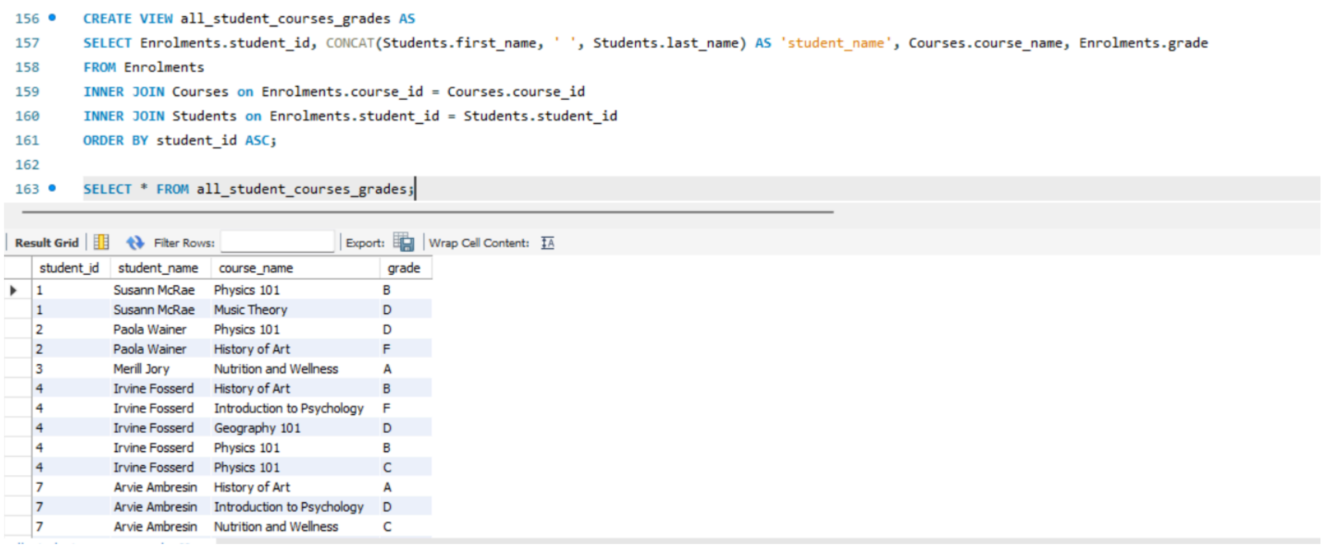
enrolment_id	student_id	course_id	enrolment_date	grade	pass_fail
8	1	3	2021-02-15	C	Pass
31	1	6	2022-02-28	A	Pass

Stored Procedure Syntax:

```sql
-- CREATE PROCEDURE procedure_name(parameter_1 parameter_1_datatype, parameter_2 parameter_2_datatye....)
-- BEGIN
-- / Code to be execued, using parameters as substitue for value to be entered when the stored procedure is called /
-- END

-- CALL procedure_name
```

View Students’ Courses and Grades

CREATE VIEW _ AS *create view to see students’ courses and grade and give it a name > SELECT FROM _ INNER JOIN ORDER BY code to be executed to show students, courses and grades >* SELECT * FROM all_student_courses_grades *to select view*



```
CREATE VIEW all_student_courses_grades AS
SELECT Enrolments.student_id, CONCAT(Students.first_name, ' ', Students.last_name) AS 'student_name', Courses.course_name, Enrolments.grade
FROM Enrolments
INNER JOIN Courses on Enrolments.course_id = Courses.course_id
INNER JOIN Students on Enrolments.student_id = Students.student_id
ORDER BY student_id ASC;

SELECT * FROM all_student_courses_grades;
```

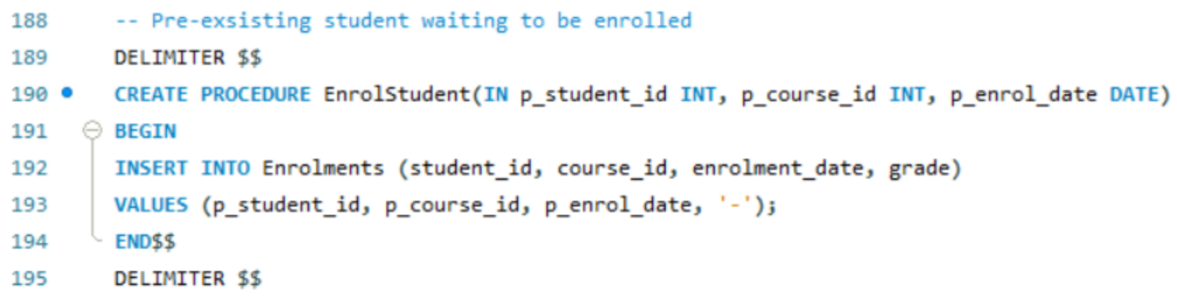
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	student_name	course_name	grade
1	Susann McRae	Physics 101	B
1	Susann McRae	Music Theory	D
2	Paola Wainer	Physics 101	D
2	Paola Wainer	History of Art	F
3	Merill Jory	Nutrition and Wellness	A
4	Irvine Fosserd	History of Art	B
4	Irvine Fosserd	Introduction to Psychology	F
4	Irvine Fosserd	Geography 101	D
4	Irvine Fosserd	Physics 101	B
4	Irvine Fosserd	Physics 101	C
7	Arvie Ambresin	History of Art	A
7	Arvie Ambresin	Introduction to Psychology	D
7	Arvie Ambresin	Nutrition and Wellness	C

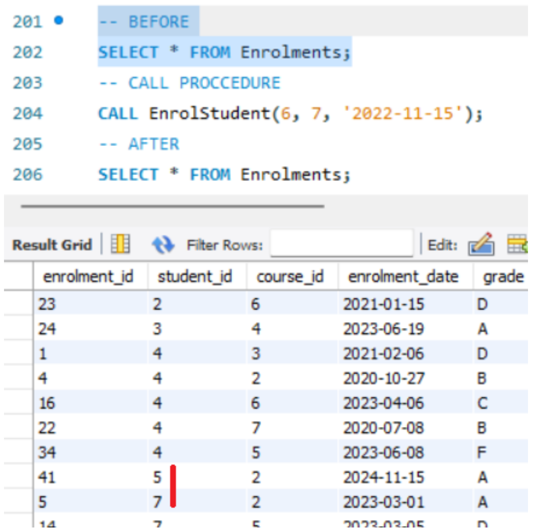
Enroll Registered Student - **Stored Procedure**

Inserting record into Enrolments *tbl i.e enrolling a student. Using* student_id *from* Students *tbl that didn’t previously exist in* Enrolments *tbl i.e registered student who is yet to enroll on a course*

DELIMITER $$ *redefining > CREATE PROCEDURE giving name and parameters < BEGIN _ END structure for procedure > INSERT INTO _ VALUES populating* Enrolments *tbl fields’ with parameters of procedure, grade left empty because student is only just enrolling*



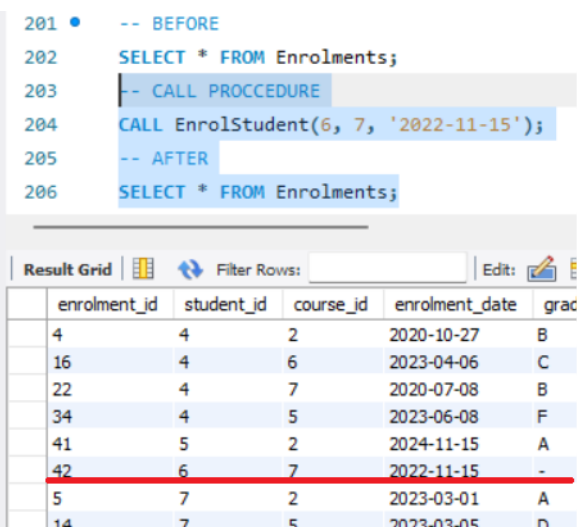
```
-- Pre-exsisting student waiting to be enrolled
DELIMITER $$
CREATE PROCEDURE EnrolStudent(IN p_student_id INT, p_course_id INT, p_enrol_date DATE)
BEGIN
INSERT INTO Enrolments (student_id, course_id, enrolment_date, grade)
VALUES (p_student_id, p_course_id, p_enrol_date, '-');
END$$
DELIMITER $$
```



```sql
-- BEFORE
SELECT * FROM Enrolments;
-- CALL PROCCEDURE
CALL EnrolStudent(6, 7, '2022-11-15');
-- AFTER
SELECT * FROM Enrolments;
```

Result Grid | Filter Rows: | Edit:

enrolment_id	student_id	course_id	enrolment_date	grade
23	2	6	2021-01-15	D
24	3	4	2023-06-19	A
1	4	3	2021-02-06	D
4	4	2	2020-10-27	B
16	4	6	2023-04-06	C
22	4	7	2020-07-08	B
34	4	5	2023-06-08	F
41	5	2	2024-11-15	A
5	7	2	2023-03-01	A
14	7	5	2023-03-05	D



```sql
-- BEFORE
SELECT * FROM Enrolments;
-- CALL PROCCEDURE
CALL EnrolStudent(6, 7, '2022-11-15');
-- AFTER
SELECT * FROM Enrolments;
```

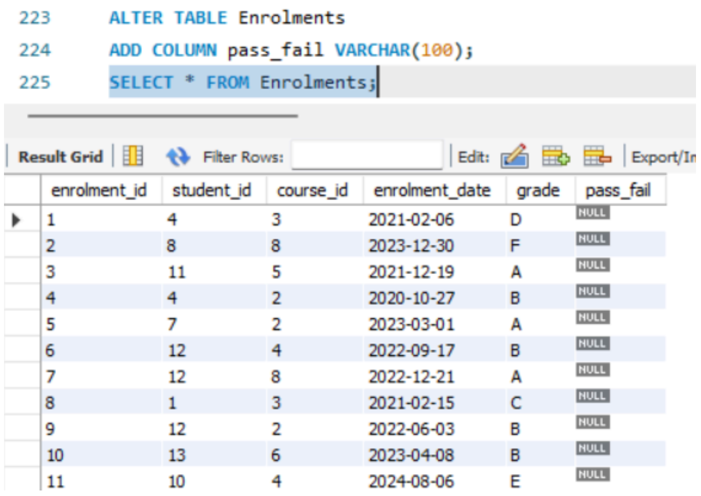
Result Grid | Filter Rows: | Edit:

enrolment_id	student_id	course_id	enrolment_date	grad
4	4	2	2020-10-27	B
16	4	6	2023-04-06	C
22	4	7	2020-07-08	B
34	4	5	2023-06-08	F
41	5	2	2024-11-15	A
42	6	7	2022-11-15	-
5	7	2	2023-03-01	A
14	7	5	2023-03-05	D

Categorise Students by Grades - **Case Statement**

Add field pass_fail *to* Enrolments *tbl to display one of three categories: pass, fail or N/A based on students'* grade value

ALTER TABLE > *indicate which table to be altered* > ADD COLUMN *and what to alter, adding a field name and datatype, which will hold category data*

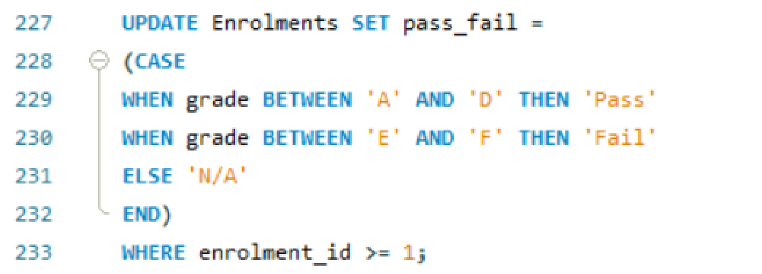


```sql
ALTER TABLE Enrolments
ADD COLUMN pass_fail VARCHAR(100);
SELECT * FROM Enrolments;
```

Result Grid | Filter Rows: | Edit: | Export/In

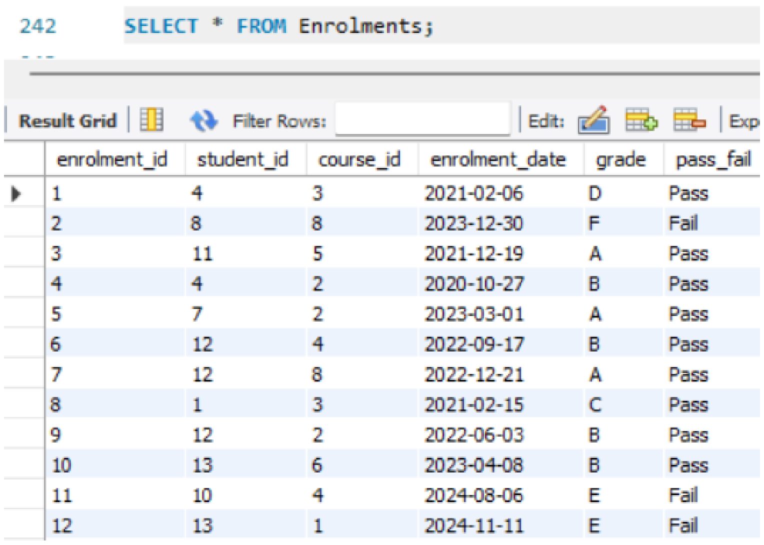
enrolment_id	student_id	course_id	enrolment_date	grade	pass_fail
1	4	3	2021-02-06	D	NULL
2	8	8	2023-12-30	F	NULL
3	11	5	2021-12-19	A	NULL
4	4	2	2020-10-27	B	NULL
5	7	2	2023-03-01	A	NULL
6	12	4	2022-09-17	B	NULL
7	12	8	2022-12-21	A	NULL
8	1	3	2021-02-15	C	NULL
9	12	2	2022-06-03	B	NULL
10	13	6	2023-04-08	B	NULL
11	10	4	2024-08-06	E	NULL

UPDATE _ SET > Enrolments *tbl to be updated and* pass_fail *values to be set* > CASE _ WHEN _ THEN _ ELSE _ END *updating multiple records based on multiple conditions e.g If the value for grade is between A and D then update* pass_fail *to reflex a category of* pass > WHERE _ *referring to condition for update to take place. All* enrolment_id's *are* >=1 *therefore this update will affect all records in* Enrolments



```sql
UPDATE Enrolments SET pass_fail =
(CASE
WHEN grade BETWEEN 'A' AND 'D' THEN 'Pass'
WHEN grade BETWEEN 'E' AND 'F' THEN 'Fail'
ELSE 'N/A'
END)
WHERE enrolment_id >= 1;
```

Resulting categories for students



```sql
SELECT * FROM Enrolments;
```

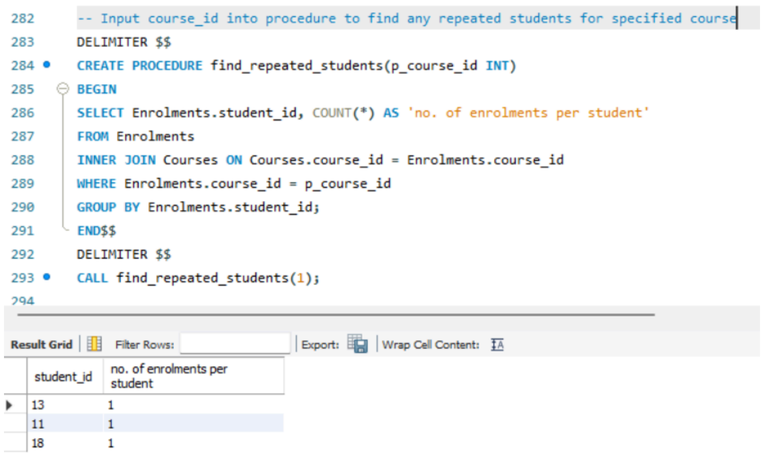
enrolment_id	student_id	course_id	enrolment_date	grade	pass_fail
1	4	3	2021-02-06	D	Pass
2	8	8	2023-12-30	F	Fail
3	11	5	2021-12-19	A	Pass
4	4	2	2020-10-27	B	Pass
5	7	2	2023-03-01	A	Pass
6	12	4	2022-09-17	B	Pass
7	12	8	2022-12-21	A	Pass
8	1	3	2021-02-15	C	Pass
9	12	2	2022-06-03	B	Pass
10	13	6	2023-04-08	B	Pass
11	10	4	2024-08-06	E	Fail
12	13	1	2024-11-11	E	Fail

Problems Encountered

Problem

with the random data generated by Mockaroo, it gave me repeated instances of student_id's that also had the same course_id's i.e duplicates of students on any given course.

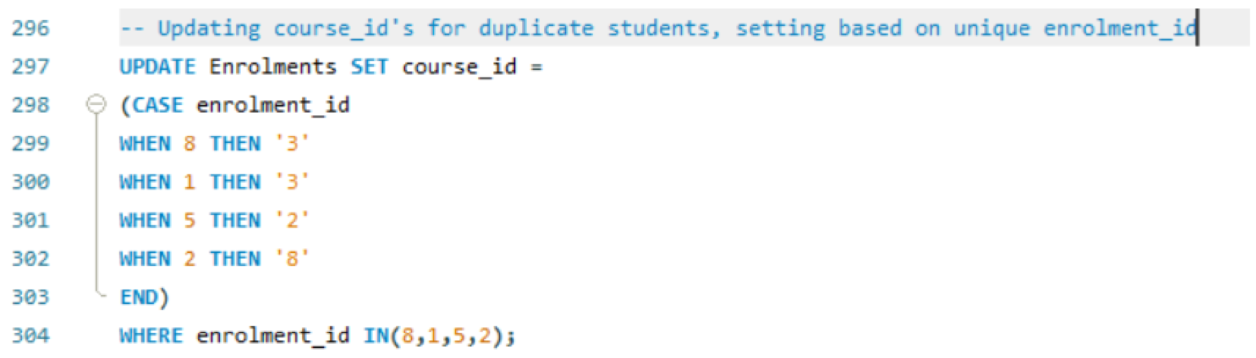
Created Procedure to find duplicates



```sql
-- Input course_id into procedure to find any repeated students for specified course
DELIMITER $$
CREATE PROCEDURE find_repeated_students(p_course_id INT)
BEGIN
SELECT Enrolments.student_id, COUNT(*) AS 'no. of enrolments per student'
FROM Enrolments
INNER JOIN Courses ON Courses.course_id = Enrolments.course_id
WHERE Enrolments.course_id = p_course_id
GROUP BY Enrolments.student_id;
END$$
DELIMITER $$
CALL find_repeated_students(1);
```

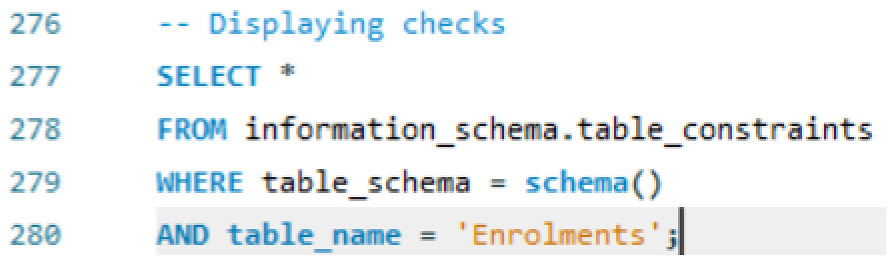
student_id	no. of enrolments per student
13	1
11	1
18	1

Created a case to update duplicates' course_id value to something different, selecting based on unique enrolment_id



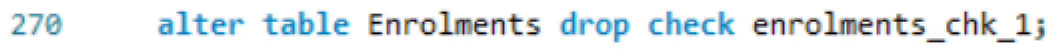
```sql
-- Updating course_id's for duplicate students, setting based on unique enrolment_id
UPDATE Enrolments SET course_id =
(CASE enrolment_id
WHEN 8 THEN '3'
WHEN 1 THEN '3'
WHEN 5 THEN '2'
WHEN 2 THEN '8'
END)
WHERE enrolment_id IN(8,1,5,2);
```

Problem
Constraint stipulated on creation of Enrolments tbl doesn't allow for '-', which I needed now that I had added a new field
Display check name, so I know what to drop



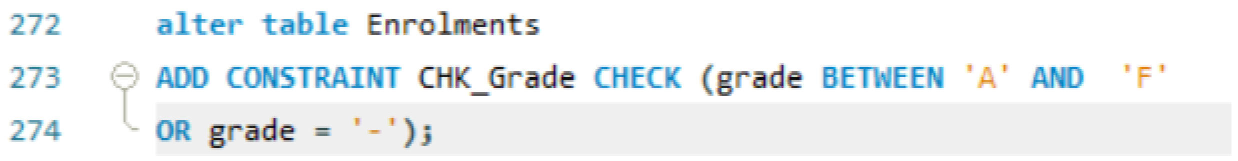
```sql
-- Displaying checks
SELECT *
FROM information_schema.table_constraints
WHERE table_schema = schema()
AND table_name = 'Enrolments';
```

Dropping the constraint



```sql
alter table Enrolments drop check enrolments_chk_1;
```

Adding a new check constraint that includes '-'



```sql
alter table Enrolments
ADD CONSTRAINT CHK_Grade CHECK (grade BETWEEN 'A' AND  'F'
OR grade = '-');
```