HW_4_DML

March 6, 2025

1 HW 4 – DDL and DML

1.1 CSPB 3287

This assignment will give you hands-on practice in working with MySQL and the DML language to manage the tables and data within the tables. You will also create Triggers that can perform additional database operations when it detects an event. In this assignment you will create new tables in your personal database on the MySQL Server. We provide some code that will connect your notebook to your database and allow you to place querys into cells in the notebook. Make sure to update the Student Information cell with your name and CU Email. Also get the connection working before trying to add solutions to the problem presented.

2 Student Information

Modify this cell and put your Name and email Name: Benjamin (Ben) Z. Chen

Email: bech1695@colorado.edu

2.1 Instructions / Notes (*Please read carefully*):

- You **may** create new Jupyter notebook cells to use for e.g. testing, debugging, exploring, etc.- this is encouraged in fact!
- However, you you must clearly mark the solution to each sub-problem by having your solution in the cell immediately after the cells marked `### BEGIN SOLUTION`
- Remember:
  - `%sql [SQL]` is for *single line* SQL queries
  - `%%sql` is for *multi line* SQL queries

```
Cell content:
    %sql [SQL query] ;
```

```
Cell content:
    %%sql
    [SQL line 1]
    [SQL line 2]
    ...
    ;
```

2.1.1 Submission Instructions:

- Do *NOT* submit your iPython notebook – instead, you should print the notebook as a PDF document and submit that. To do that, use `File -> Export Notebook As.. -> HTML`, then open the HTML document and print it to a PDF file.

If you run into problems with a query taking a very very long time, first try `Kernel -> Restart All and Run All Cells..` and then ask on Piazza

Have fun!

2.1.2 Configuring your access to MySQL Server

You need to configure the system to access the shared MySQL Server. To access the server from the Jupyter Notebook we have provided some code that sets up the access. You will use a SQL magic library that allows your to insert SQL calls into the cells of a notebook.

[1]:
```
#
# Read the configuration file into an object so we can extract attributes
# to use to setup the connection to the MySQL Server.
#
import os
import configparser

config_file_path = "./mysql.cfg"

mycfg = configparser.ConfigParser()
mycfg.read(config_file_path)                          # reads the configuration␣
 ↪file from the parent directory to where this notebook resides

# override the user and passwd with credentials (delete this part if you want␣
 ↪to use test login)
mycfg['mysql']['user'] = 'bech1695'
mycfg['mysql']['passwd'] = '847fa06b9fd42e2d30ae'

print(f"User    : [{mycfg['mysql']['user']}]")  # verify the data from the␣
 ↪config file

database = mycfg['mysql']['user']                # by default you will want to␣
 ↪access your personal database on the server
                                                 # other times you will want to␣
 ↪access a shared database (such as the Northwind database)
print(f"Database: [[mysql://{mycfg['mysql']['user']}...@{database}]")  # print␣
 ↪info to the display but without the password

# build the URL to access the database.  This includes the user name, password,␣
 ↪and database to be used
```

```
db_url = f"mysql://{mycfg['mysql']['user']}:
 ↪{mycfg['mysql']['passwd']}@applied-sql.cs.colorado.edu:3306/
 ↪{mycfg['mysql']['user']}"

os.environ['DATABASE_URL'] = db_url              # Set the URL in the System␣
 ↪Environment for use by SQL magic
```

```
User     : [bech1695]
Database: [[mysql://bech1695…@bech1695]
```

Get the extension to handle SQL magic and Verify the Access to the Database The SQL module will all the use of `%%` and `%%sql` commands within the cells of the notebook to access the database. The `%sql SELECT version()` command will send the SELECT statement to the database. Here we are asking for the current version number of the server software. If the command is successful in reaching the server, it will list out the version number (8.0.33 at last update to this document).

[2]:
```
%reload_ext sql
print ("get version...")
%sql SELECT version()
```

```
get version…
1 rows affected.
```

[2]:
```
[('8.0.33',)]
```

[3]:
```
%sql SELECT * FROM Assessments;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
(MySQLdb.ProgrammingError) (1146, "Table 'bech1695.Assessments' doesn't exist")
[SQL: SELECT * FROM Assessments;]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

[4]:
```
# I don't have an assessments table, the assessments table should be from the␣
 ↪test 3287 login, which shows the assessments in the class
```

2.2 Constraints Review

Constraints are tools to impose restrictions on allowable data within a database, beyond the requirements imposed by table definition types.

Constraints, also known as *integrity constraints*, are used to constrain allowable database states and have the DBMS validate those rules. They prevent disallowed values from being entered into the database. Common uses of constraints are listed below. * non-null constraints * `create Table MyTable(myValue dataType NOT NULL);` * key or uniqueness constraints * `create Table MyTable(myId int PRIMARY KEY);` * `create Table MyTable(myValue1 dataType, myValue2 dataType, UNIQUE(myValue1,myValue2));` *

attribute restrictions * `create Table MyTable(myValue dataType check(myValue > 0))` * referential integrity (a.k.a. foreign keys) * `create Table MyTable(otherId int, foreign key(otherId) references OtherTable(otherColumn))`

Write your table definitions here. **Format your definitions so they are readable.**

[5]:
```
### BEGIN SAMPLE SOLUTION
```

[6]:
```
%%sql
-- comment Drop table before Creating newest version
DROP TABLE IF EXISTS tutorials_tbl;
CREATE TABLE IF NOT EXISTS
      tutorials_tbl
        (
          tutorial_id INT NOT NULL AUTO_INCREMENT,
          tutorial_title VARCHAR(100) NOT NULL,
          tutorial_author VARCHAR(40) NOT NULL,
          submission_date DATE,
          PRIMARY KEY ( tutorial_id )
        ) ;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
0 rows affected.
```

[6]:
```
[]
```

[7]:
```
### END SAMPLE SOLUTION
```

The code above should have created (or recreated) a table named "tutorials_tbl" in your personal database on the MySQL Server. Use your GUI interface to access your database and verify you have created the table.

3 Now you can begin your solutions to the problems listed below

Remember to place your solution between the BEGIN and END cells.

[8]:
```
### BEGIN SOLUTION
```

[9]:
```
# my code to test if tutorials_tbl was added

%sql SHOW TABLES;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
11 rows affected.
```

[9]:
```
[('Class',),
 ('ClassGrade',),
```

```
('Customer',),
('Department',),
('Employee',),
('Item',),
('Notification',),
('Retail_Sale',),
('Retail_Sale_Item',),
('Student',),
('tutorials_tbl',)]
```

[10]:
```
### END SOLUTION
```

can use:

%%sql

DROP TABLE IF EXISTS ClassGrade; DROP TABLE IF EXISTS Notification; DROP TABLE IF EXISTS Class; DROP TABLE IF EXISTS Student;

to drop existing tables

3.1 Question 1 - Creating Tables with Constraints [20 pts]

Write CREATE TABLE declarations with the necessary constraints for the following 4 tables and their specifications:

- `Student(sID, name, parentEmail, gpa)`
  - `sID (should be unique)`
  - `name (should exist)`
  - `parentEmail(should exist)`
  - `gpa (real value between 0 and 4 inclusive)`
- `Class(cID, name, units)`
  - `cID (should be unique and always exactly 8 characters)`
  - `name (should exist)`
  - `units (must be between 1 and 5 inclusive)`
- `ClassGrade(sID, cID, grade)`
  - `sID (should reference a student)`
  - `cID (should reference a class)`
  - `grade (integer between 0 and 4 inclusive, for F,D,C,B,A)`
  - `make sure that a student can only get 1 grade for each class`
- `Notification(parentEmail, text)`
  - `parentEmail (should exist)`
  - `text (the message body, should exist)`

Constraints, such as the value for `grade`, **must** use `check` to check that constraint.
You will probably want to `DROP` the table if it exists *before* trying to create the table.
The `CREATE` will fail if the table is already defined. See the MySQL Documentation for DROP

[11]:
```
### BEGIN SOLUTION
```

[12]:
```
%%sql

-- Drop in reverse dependency order
DROP TABLE IF EXISTS ClassGrade;
DROP TABLE IF EXISTS Notification;
DROP TABLE IF EXISTS Class;
DROP TABLE IF EXISTS Student;

-- ---------------------------
-- 1) Student(sID, name, parentEmail, gpa)
--    - sID should be unique (PRIMARY KEY here)
--    - name, parentEmail not null
--    - gpa between 0 and 4 inclusive
-- ---------------------------
CREATE TABLE Student (
    sID INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    parentEmail VARCHAR(100) NOT NULL,
    gpa DECIMAL(2,1) NOT NULL,
    CHECK (gpa >= 0 AND gpa <= 4),
    PRIMARY KEY (sID)
);

-- ---------------------------
-- 2) Class(cID, name, units)
--    - cID unique (PRIMARY KEY), exactly 8 chars
--    - name not null
--    - units between 1 and 5 inclusive
-- ---------------------------
CREATE TABLE Class (
    cID CHAR(8) NOT NULL,
    name VARCHAR(100) NOT NULL,
    units INT NOT NULL,
    CHECK (units >= 1 AND units <= 5),
    PRIMARY KEY (cID)
);

-- ---------------------------
-- 3) ClassGrade(sID, cID, grade)
--    - sID references Student(sID)
--    - cID references Class(cID)
--    - grade int between 0 and 4 inclusive (0=F ... 4=A)
--    - a student can only get 1 grade for each class (PRIMARY KEY on sID + cID)
-- ---------------------------
CREATE TABLE ClassGrade (
    sID INT NOT NULL,
    cID CHAR(8) NOT NULL,
```

```
    grade INT NOT NULL,
    CHECK (grade >= 0 AND grade <= 4),
    PRIMARY KEY (sID, cID),
    FOREIGN KEY (sID) REFERENCES Student(sID),
    FOREIGN KEY (cID) REFERENCES Class(cID)
);

-- ----------------------------
-- 4) Notification(parentEmail, text)
--    - parentEmail should exist (NOT NULL)
--    - text (the message) should exist (NOT NULL)
-- ----------------------------
CREATE TABLE Notification (
    parentEmail VARCHAR(100) NOT NULL,
    text VARCHAR(255) NOT NULL
    -- Optionally, if we want to enforce that this matches a
    -- Student's parentEmail, we can use foreign key:
    -- FOREIGN KEY (parentEmail) REFERENCES Student(parentEmail)
    -- that'll require parentEmail to be UNIQUE or indexed in Student.
);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

[12]:
```
[]
```

[13]:
```
### END SOLUTION
```

[14]:
```
%sql SHOW TABLES;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
11 rows affected.
```

[14]:
```
[('Class',),
 ('ClassGrade',),
 ('Customer',),
 ('Department',),
 ('Employee',),
 ('Item',),
 ('Notification',),
```

```
 ('Retail_Sale',),
 ('Retail_Sale_Item',),
 ('Student',),
 ('tutorials_tbl',)]
```

3.2 Question 2 - Validate Tables were Created Correctly [10 pts]

Write SQL queries to validate the information stored in the database for each of the tables created.

As you write code that has constraints or triggers, you will need to verify that your code is working as expected. This means you should write tests that validate each of the conditions you have specified in the constraints.

The easiest method for these tests is to attempt to insert a row with one of the parameters being invalid. For example, if an integer column that is representing a year value is being tested, you would need to try values that are valid and invalid. If the year must be in YYYY format, any value less than 1000 would be invalid. If you added further constraints (such as after 1970) then you would test that a value like 1942 causes an error and that 1982 does not.

You can issue a SQL query to insert a new row and give it an invalid argument like the invalid `name` used below. It will generate an exception and ultimately tell you that the query is invalid because of the NULL name.

```
    INSERT INTO Students VALUES(1, NULL, "somewhere@gmail", 0.0)
```

You will receive a lengthy error message about the exception.

```
---------------------------------------------------------------------------
IntegrityError                                Traceback (most recent call last)
File /opt/conda/lib/python3.10/site-packages/sqlalchemy/engine/base.py:1900, in Connection._exe
   1899     if not evt_handled:
-> 1900         self.dialect.do_execute(
   1901             cursor, statement, parameters, context
   1902         )
.
.
```

Eventually you will see the exception is because of the NULL name.

```
.
.
IntegrityError: (MySQLdb.IntegrityError) (1048, "Column 'name' cannot be null")
[SQL: INSERT INTO Student VALUES (9, NULL, "99-99", 1.);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
```

In Python you can use the `try - except` mechanism to capture the exceptions and print only the meaningful information.

```
## try name as NULL
try:
    %sql INSERT INTO Student VALUES (9, NULL, "99-99", 1.);
except Exception as e:
```

```
    print ("Testing name as NULL:", e)
```

Instead of the full traceback from the previous query, you get the following:

```
 * mysql://knoxd:***@applied-sql.cs.colorado.edu:3306/knoxd
Testing name as NULL: (MySQLdb.IntegrityError) (1048, "Column 'name' cannot be null")
[SQL: INSERT INTO Student VALUES (9, NULL, "99-99", 1.);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
```

You will need to create a `try - except` for each test because if there are multiple fields that are tested in one `try` statement, the first to cause an exception will be caught, but the rest of the testing code will be skipped. The general format of a cell should look like the cell below.

[15]:
```
 ## Testing that NULL title will cause an exception

try:
    %sql INSERT INTO tutorials_tbl VALUES (0, NULL, "Knuth", NULL)
except Exception as e:
    print ("Testing TITLE as NULL:", e)
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing TITLE as NULL: (MySQLdb.IntegrityError) (1048, "Column 'tutorial_title'
cannot be null")
[SQL: INSERT INTO tutorials_tbl VALUES (0, NULL, "Knuth", NULL)]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
```

3.2.1 Create a set of tests that will verify that each of your constraints in each table (Student, Class, ClassGrade, Notification) are behaving correctly.

[16]:
```
### BEGIN SOLUTION
```

[17]:
```
# We should test each of the constraints, and test when they do and do not work

### Student Constraints:
###  - sID must be unique
###  - name NOT NULL
###  - parentEmail NOT NULL
###  - gpa between 0 and 4 inclusive

# 1) Test insert Student with NULL name
try:
    %sql INSERT INTO Student VALUES (100, NULL, 'pEmail@example.com', 2.5);
except Exception as e:
    print("Testing Student name is NULL:", e)

# 2) Test insert Student with NULL parentEmail
try:
    %sql INSERT INTO Student VALUES (101, 'Alice', NULL, 3.0);
except Exception as e:
```

```
    print("Testing Student parentEmail is NULL:", e)

# 3) Test insert Student with invalid gpa > 4
try:
    %sql INSERT INTO Student VALUES (102, 'Bob', 'bob@example.com', 4.1);
except Exception as e:
    print("Testing Student gpa > 4.0:", e)

# 4) Test insert valid row
try:
    %sql INSERT INTO Student VALUES (103, 'Carol', 'carol@example.com', 3.8);
    print("Valid Student inserted (sID=103, gpa=3.8).")
except Exception as e:
    print("Testing Unexpected Failure on valid Student row:", e)

# 5) Test insert duplicate sID=103 to test uniqueness
try:
    %sql INSERT INTO Student VALUES (103, 'Carol2', 'carol2@example.com', 3.0);
except Exception as e:
    print("Testing Student sID must be unique:", e)

# This looks right because the output shows constraint violations with NOT␣
 ↪NULL, CHECK, and uniqueness, and each
# trigger an error message, confirming that the database is enforcing the␣
 ↪constrains and Valid Insert works without
# an exception
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing Student name is NULL: (MySQLdb.IntegrityError) (1048, "Column 'name'
cannot be null")
[SQL: INSERT INTO Student VALUES (100, NULL, 'pEmail@example.com', 2.5);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing Student parentEmail is NULL: (MySQLdb.IntegrityError) (1048, "Column
'parentEmail' cannot be null")
[SQL: INSERT INTO Student VALUES (101, 'Alice', NULL, 3.0);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
(MySQLdb.OperationalError) (3819, "Check constraint 'Student_chk_1' is
violated.")
[SQL: INSERT INTO Student VALUES (102, 'Bob', 'bob@example.com', 4.1);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
Valid Student inserted (sID=103, gpa=3.8).
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing Student sID must be unique: (MySQLdb.IntegrityError) (1062, "Duplicate
```

```
entry '103' for key 'Student.PRIMARY'")
[SQL: INSERT INTO Student VALUES (103, 'Carol2', 'carol2@example.com', 3.0);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
```

[18]:
```
# Class Table Constraints:
# cID (unique, exactly 8 characters)
# name (NOT NULL)
# units (1 ... 5 inclusive)

# 1) Test class with cID shorter than 8 chars
try:
    %sql INSERT INTO Class VALUES ('ABC123', 'ShortID', 3);
except Exception as e:
    print("Testing cID < 8 chars:", e)

# 2) Test class with NULL name
try:
    %sql INSERT INTO Class VALUES ('CLASS111', NULL, 4);
except Exception as e:
    print("Testing Class name NULL:", e)

# 3) Test class with units > 5
try:
    %sql INSERT INTO Class VALUES ('CLASS222', 'History', 6);
except Exception as e:
    print("Testing Class units > 5:", e)

# 4) Test valid row
try:
    %sql INSERT INTO Class VALUES ('CLASS333', 'Algebra II', 4);
    print("Valid Class inserted (CLASS333).")
except Exception as e:
    print("Testing Unexpected Failure on valid Class row:", e)

# 5) Test insert a second time with same cID to test uniqueness
try:
    %sql INSERT INTO Class VALUES ('CLASS333', 'Physics', 3);
except Exception as e:
    print("Testing cID must be unique:", e)

# This tests for invalid inserts failing with an erorr (showing the constraints␣
 ↪were enforced), and valid inserts are ok,
# showing "1 rows affected" which confirms a new row was inserted properly
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing Class name NULL: (MySQLdb.IntegrityError) (1048, "Column 'name' cannot
```

```
be null")
[SQL: INSERT INTO Class VALUES ('CLASS111', NULL, 4);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
(MySQLdb.OperationalError) (3819, "Check constraint 'Class_chk_1' is violated.")
[SQL: INSERT INTO Class VALUES ('CLASS222', 'History', 6);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
Valid Class inserted (CLASS333).
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing cID must be unique: (MySQLdb.IntegrityError) (1062, "Duplicate entry
'CLASS333' for key 'Class.PRIMARY'")
[SQL: INSERT INTO Class VALUES ('CLASS333', 'Physics', 3);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
```

[19]:
```
# ClassGrade Table Constraints:

# sID (FK -> Student.sID)
# cID (FK -> Class.cID)
# grade (0 ... 4 inclusive)
# (sID, cID) = primary key -> only 1 grade per class for each student

# 1) Insert into ClassGrade referencing non-existent Student
try:
    %sql INSERT INTO ClassGrade VALUES (9999, 'CLASS333', 3);
except Exception as e:
    print("Testing ClassGrade with invalid sID:", e)

# 2) Insert into ClassGrade referencing non-existent Class
try:
    %sql INSERT INTO ClassGrade VALUES (103, 'BADCLASS', 2);
except Exception as e:
    print("Testing ClassGrade with invalid cID:", e)

# 3) Insert ClassGrade with invalid grade > 4
try:
    %sql INSERT INTO ClassGrade VALUES (103, 'CLASS333', 5);
except Exception as e:
    print("Testing ClassGrade.grade > 4:", e)

# 4) Valid ClassGrade (assuming sID=103 exists in Student, CLASS333 exists in␣
 ↪Class)
try:
    %sql INSERT INTO ClassGrade VALUES (103, 'CLASS333', 4);
    print("Valid ClassGrade inserted (sID=103, CLASS333).")
except Exception as e:
```

```
    print("Testing Unexpected Failure on valid ClassGrade row:", e)

# 5) Attempt duplicate (sID=103, cID=CLASS333)
try:
    %sql INSERT INTO ClassGrade VALUES (103, 'CLASS333', 2);
except Exception as e:
    print("Testing (sID, cID) must be unique in ClassGrade:", e)

# We see errors for invalid inserts (foreign key references that don't exist,␣
 ↪grade out of range, duplicate primary key, etc.)
# and see "1 rows affected" for valid inserts
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing ClassGrade with invalid sID: (MySQLdb.IntegrityError) (1452, 'Cannot add
or update a child row: a foreign key constraint fails (`bech1695`.`ClassGrade`,
CONSTRAINT `ClassGrade_ibfk_1` FOREIGN KEY (`sID`) REFERENCES `Student`
(`sID`))')
[SQL: INSERT INTO ClassGrade VALUES (9999, 'CLASS333', 3);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing ClassGrade with invalid cID: (MySQLdb.IntegrityError) (1452, 'Cannot add
or update a child row: a foreign key constraint fails (`bech1695`.`ClassGrade`,
CONSTRAINT `ClassGrade_ibfk_2` FOREIGN KEY (`cID`) REFERENCES `Class` (`cID`))')
[SQL: INSERT INTO ClassGrade VALUES (103, 'BADCLASS', 2);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
(MySQLdb.OperationalError) (3819, "Check constraint 'ClassGrade_chk_1' is
violated.")
[SQL: INSERT INTO ClassGrade VALUES (103, 'CLASS333', 5);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
Valid ClassGrade inserted (sID=103, CLASS333).
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing (sID, cID) must be unique in ClassGrade: (MySQLdb.IntegrityError) (1062,
"Duplicate entry '103-CLASS333' for key 'ClassGrade.PRIMARY'")
[SQL: INSERT INTO ClassGrade VALUES (103, 'CLASS333', 2);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
```

[20]:
```
# Notification Table Constraints:

# parentEmail (NOT NULL)
# text (NOT NULL)

# 1) Insert Notification with NULL parentEmail
try:
    %sql INSERT INTO Notification VALUES (NULL, 'Hello!');
```

```
except Exception as e:
    print("Testing Notification parentEmail is NULL:", e)

# 2) Insert Notification with NULL text
try:
    %sql INSERT INTO Notification VALUES ('pEmail@example.com', NULL);
except Exception as e:
    print("Testing Notification text is NULL:", e)

# 3) Valid Notification
try:
    %sql INSERT INTO Notification VALUES ('pEmail@example.com', 'Child is doing␣
 ↪well.');
    print("Valid Notification inserted.")
except Exception as e:
    print("Unexpected Failure on Notification even though it's valid", e)

# Attempted insert with parentEmail failed, confirms databased refused insert.␣
 ↪Attempted insert with text also failed,
# raised the same type of error. Then 1 rows affected and inserted something␣
 ↪valid to the Notifications Table
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing Notification parentEmail is NULL: (MySQLdb.IntegrityError) (1048,
"Column 'parentEmail' cannot be null")
[SQL: INSERT INTO Notification VALUES (NULL, 'Hello!');]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
Testing Notification text is NULL: (MySQLdb.IntegrityError) (1048, "Column
'text' cannot be null")
[SQL: INSERT INTO Notification VALUES ('pEmail@example.com', NULL);]
(Background on this error at: https://sqlalche.me/e/14/gkpj)
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
Valid Notification inserted.
```

[21]:
```
### END SOLUTION
```

[22]:
```
%%sql

SHOW TABLES;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
11 rows affected.
```

[22]:
```
[('Class',),
 ('ClassGrade',),
```

```
('Customer',),
('Department',),
('Employee',),
('Item',),
('Notification',),
('Retail_Sale',),
('Retail_Sale_Item',),
('Student',),
('tutorials_tbl',)]
```

[23]:
```
# just wanted to see if the tables were still there
```

3.3 Question 3 - Populate each of the Tables with Data [10 pts]

Write SQL queries to add items to each of the tables. Add separate cells within the solution to **show the rows of the table before and after the insertion.** You need to make sure table is empty before insertion.

HINT: three cells at least, (1) to show tables empty, (2) to insert rows, (3) to show values in the tables after insertions.

Add the following Courses, Students, ClassGrades:

- Courses:
  - the CSPB 3287 is 3 credits
  - the CSPB 3308 is 3 credits
  - the CSPB 3753 is 4 credits
  - the CSPB 4122 is 3 credits
- Student id, name, email:
  - 1, Taylor, MorgansMom@gmail.com
  - 2, Jordon, JordonsGram@gmail.com
  - 3, Riley, RileysDad@gmail.com
- Course Grades
  - Taylor got a B in 3308, A in 3287
  - Jordon got a A in 3308, B in 3753
  - Taylor got a C in 4122
- Notification(parentEmail, text)
  - Add message "Riley needs to take some courses" that was sent to RileysDad@gmail.com

[24]:
```
### BEGIN SOLUTION
```

[25]:
```
%%sql

-- Remove old data to start fresh
DELETE FROM ClassGrade;
DELETE FROM Notification;
DELETE FROM Student;
DELETE FROM Class;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
1 rows affected.
2 rows affected.
```

[25]:
```
[]
```

[26]:
```
%%sql
SELECT 'Class Table' AS TableName, Class.*
FROM Class;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
```

[26]:
```
[]
```

[27]:
```
%%sql
SELECT 'Student Table' AS TableName, Student.*
FROM Student;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
```

[27]:
```
[]
```

[28]:
```
%%sql
SELECT 'ClassGrade Table' AS TableName, ClassGrade.*
FROM ClassGrade;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
```

[28]:
```
[]
```

[29]:
```
%%sql
SELECT 'Notification Table' AS TableName, Notification.*
FROM Notification;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
```

[29]:
```
[]
```

[30]:
```
# If the table is truly empty, the result will show 0 rows returned. That␣
 ↪confirms any old data was successfully removed from the table.
```

[31]:
```
%%sql

-- ------------------------------
-- Courses (cID must be 8 chars)
-- ------------------------------
INSERT INTO Class VALUES('CSPB3287', 'CSPB 3287', 3);
INSERT INTO Class VALUES('CSPB3308', 'CSPB 3308', 3);
INSERT INTO Class VALUES('CSPB3753', 'CSPB 3753', 4);
INSERT INTO Class VALUES('CSPB4122', 'CSPB 4122', 3);

-- ------------------------------
-- Students (sID, name, parentEmail, gpa)
-- ------------------------------
INSERT INTO Student VALUES (1, 'Taylor', 'MorgansMom@gmail.com', 0.0);
INSERT INTO Student VALUES (2, 'Jordon', 'JordonsGram@gmail.com', 0.0);
INSERT INTO Student VALUES (3, 'Riley', 'RileysDad@gmail.com', 0.0);

-- ------------------------------
-- Course Grades:
--   Taylor got a B=3 in 3308, A=4 in 3287, C=2 in 4122
--   Jordon got an A=4 in 3308, B=3 in 3753
-- ------------------------------
INSERT INTO ClassGrade VALUES (1, 'CSPB3308', 3);
INSERT INTO ClassGrade VALUES (1, 'CSPB3287', 4);
INSERT INTO ClassGrade VALUES (1, 'CSPB4122', 2);
INSERT INTO ClassGrade VALUES (2, 'CSPB3308', 4);
INSERT INTO ClassGrade VALUES (2, 'CSPB3753', 3);

-- ------------------------------
-- Notification:
--   "Riley needs to take some courses" -> RileysDad@gmail.com
-- ------------------------------
INSERT INTO Notification VALUES('RileysDad@gmail.com', 'Riley needs to take␣
 ↪some courses');
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

```
1 rows affected.
1 rows affected.
```

[31]:
```
[]
```

[32]:
```
# should be 4 rows in class: CSPB3287, CSPB3308, CSPB3753, CSPB4122
# 3 rows in student: Taylor, Jordon, and Riley
# 5 rows in ClassGrade: Taylor's B, A, C, and Jordon's A, B
# 1 row in Notification to Riley's Dad
```

[33]:
```
%%sql

SELECT 'Class Table' AS TableName, Class.*
FROM Class;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
4 rows affected.
```

[33]:
```
[('Class Table', 'CSPB3287', 'CSPB 3287', 3),
 ('Class Table', 'CSPB3308', 'CSPB 3308', 3),
 ('Class Table', 'CSPB3753', 'CSPB 3753', 4),
 ('Class Table', 'CSPB4122', 'CSPB 4122', 3)]
```

[34]:
```
%%sql

SELECT 'Student Table' AS TableName, Student.*
FROM Student;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
3 rows affected.
```

[34]:
```
[('Student Table', 1, 'Taylor', 'MorgansMom@gmail.com', Decimal('0.0')),
 ('Student Table', 2, 'Jordon', 'JordonsGram@gmail.com', Decimal('0.0')),
 ('Student Table', 3, 'Riley', 'RileysDad@gmail.com', Decimal('0.0'))]
```

[35]:
```
%%sql

SELECT 'ClassGrade Table' AS TableName, ClassGrade.*
FROM ClassGrade;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
5 rows affected.
```

[35]:
```
[('ClassGrade Table', 1, 'CSPB3287', 4),
 ('ClassGrade Table', 1, 'CSPB3308', 3),
 ('ClassGrade Table', 1, 'CSPB4122', 2),
 ('ClassGrade Table', 2, 'CSPB3308', 4),
 ('ClassGrade Table', 2, 'CSPB3753', 3)]
```

[36]:
```
%%sql

SELECT 'Notification Table' AS TableName, Notification.*
FROM Notification;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[36]:
```
[('Notification Table', 'RileysDad@gmail.com', 'Riley needs to take some
courses')]
```

[37]:
```
%%sql

SELECT *
FROM Student;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
3 rows affected.
```

[37]:
```
[(1, 'Taylor', 'MorgansMom@gmail.com', Decimal('0.0')),
 (2, 'Jordon', 'JordonsGram@gmail.com', Decimal('0.0')),
 (3, 'Riley', 'RileysDad@gmail.com', Decimal('0.0'))]
```

[38]:
```
# now it shows a table or the student's name, their sID, parentEmail, and the 0.
 ↪0 gpa
```

[39]:
```
### END SOLUTION
```

3.4 Question 4 - Modify the Table Structure [30 pts]

In a separate cells within the solution, write query to handle the modifications listed below.

1. School administration wants to know if the notifications are useful. A new field must be added to remember a rating that can be set by the recipient. Update the `Notification` table to place a new field called `rating` to store an integer value between 1 and 5. A value of 0 should be the default value for rating indicating there is not a value set.
2. Set the values in the `Notification` table for `rating` to a random value between 0 and 5. See MySQL Documentation for RAND() function.
3. Show the notifications table with the newly added random data. List the notifications in descending order of customer satisfaction.

[40]:
```
### BEGIN SOLUTION 4.1
```

[41]:
```
# Random sampling: ORDER BY RAND() to randomly order rows (though note␣
 ↪performance implications for large tables). Generating random data in tests␣
 ↪or dummy columns.
# Seeding a PRNG sequence for repeatable testing (using RAND(<seed>))
```

[42]:
```
%%sql
ALTER TABLE Notification
  ADD COLUMN rating INT NOT NULL DEFAULT 0
  CHECK (rating >= 0 AND rating <= 5);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[42]: []

[43]:
```
### END SOLUTION 4.1
```

[44]:
```
### BEGIN SOLUTION 4.2
```

[45]:
```
%%sql
UPDATE Notification
SET rating = FLOOR(RAND() * 6);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[45]: []

[46]:
```
# ran this cell twice to make sure the ratings update actually like a random␣
 ↪number
# RAND() * 6 is not inclusive of 6, it's <6
```

[47]:
```
### END SOLUTION 4.2
```

[48]:
```
### BEGIN SOLUTION 4.3
```

[49]:
```
%%sql
SELECT *
FROM Notification
ORDER BY rating DESC;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[49]: [('RileysDad@gmail.com', 'Riley needs to take some courses', 1)]

[50]:
```
### END SOLUTION 4.3
```

4 Triggers Introduction

Triggers are used to execute sql commands upon changes to the specified tables.
Documentation on Trigger support in SQLite can be found here.

Notice that there are no constraints on the tables being created. This can cause problems later and your tables should always have contraints. However they are not requred for our example and we have left them off.

[51]:
```
%%sql
DROP TABLE IF EXISTS Employee;
DROP TABLE IF EXISTS Department;
DROP TRIGGER IF EXISTS update_employee_count;
CREATE TABLE Employee(eID int, name text, dID int);
CREATE TABLE Department(dID int, name text, employee_count int);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

[51]:
```
[]
```

[52]:
```
%%sql
CREATE TRIGGER update_employee_count
AFTER INSERT on Employee
FOR EACH ROW
BEGIN
  UPDATE Department SET employee_count = employee_count + 1
    WHERE dID = new.dID;
END;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
```

[52]:
```
[]
```

4.0.1 Note that there is a difference between OLD values and NEW values in triggers that execute on statements that change values in a table.

Both the WHEN clause and the trigger actions may access elements of the row being inserted, deleted or updated using references of the form "NEW.column-name" and "OLD.column-name", where column-name is the name of a column from the table that the trigger is associated with. Triggers on INSERT statements (like that above) can only access the NEW values (since OLD values don't exist!) and triggers on DELETE statements can only access OLD values.

Let's continue by adding data to the tables.

[53]:
```
%%sql
INSERT INTO Department VALUES(1,'HR',0);
INSERT INTO Department VALUES(2,'Engineering',0);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
```

[53]: []

At this point, there are no empoloyees in the Employee table. As you can see below, each department has 0 employees.

[54]:
```
%%sql
SELECT name, employee_count
FROM Department;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[54]: [('HR', 0), ('Engineering', 0)]

When we insert several employees into the Employee table, the trigger should fire and update values in the Department table.

[55]:
```
%%sql
-- values are eID, name, dID
INSERT INTO Employee VALUES
    (1,'Taylor',1),
    (2,'Jordon',1),
    (3,'Riley',2)
    ;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
3 rows affected.
```

[55]: []

Now when we view the employee table, we see that the employee count has been updated by the trigger.

[56]:
```
%%sql
SELECT name, employee_count
FROM Department;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[56]: [('HR', 2), ('Engineering', 1)]

4.1 Question 5 - Generate the GPA for each Students' Grades [10 pts]

Below in Question 6, you will write a Trigger to update field of one table when data inserted into another table. You will need to write a Trigger to update the GPA for a student when a new grade is added to the ClassGrades table.

But first, you will need to first figure out how to calculate the GPA from the data in the ClassGrades table. Once you have a query to generate the GPA, you can use it as a subquery in other queries. Here you need to JOIN the ClassGrades with the course information about number of credits.

The GPA of each student is calculated based on the grades and number of units of the courses.

```
gpa = sum(units*grade)/sum(units)
```

[57]:
```
%%sql

select * from student;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
(MySQLdb.ProgrammingError) (1146, "Table 'bech1695.student' doesn't exist")
[SQL: select * from student;]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

The calculation is straight forward when only one course grade is entered. Make sure you can correctly calculate the GPA before trying to have a trigger update the value.

[58]:
```
### BEGIN SOLUTION
```

[59]:
```
%%sql

SELECT
    s.sID,
    s.name,
    -- sum of (units * numericGrade) divided by sum of units
    SUM(c.units * cg.grade) / SUM(c.units) AS computed_gpa
FROM Student AS s
JOIN ClassGrade AS cg
    ON s.sID = cg.sID
JOIN Class AS c
    ON cg.cID = c.cID
GROUP BY s.sID, s.name;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[59]:
```
[(1, 'Taylor', Decimal('3.0000')), (2, 'Jordon', Decimal('3.4286'))]
```

[60]:
```
# SUM(c.units * cg.grade) / SUM(c.units) AS computed_gpa is the GPA logic on
  ↪how GPA is calculated in school
# SUM(c.units) sums the total course units the student has taken.
```

```
# if we only need ID and GPA, we can omit s.name
# if any student has no ClassGrade rows, the student won't show up when we␣
 ↪query the result unless we do some kind of join like outer join or handle␣
 ↪divisions by 0

# these scores are correct and can be confirmed by plugging them in, Taylor =␣
 ↪(9 + 12 + 6)/9 = 3.0 and Jordon is (12 + 12)/7 = roughly 3.4286
```

[61]:
```
### END SOLUTION
```

The calculation is straight forward when only one course grade is entered. Make sure you can correctly calculate the GPA with multiple courses and grades before trying to have a trigger update the value.

Add some more grades so you know it works with 1, 2, 3, … grades.

Test your calculation again to make sure it still works with more grades.

[62]:
```
### BEGIN SOLUTION
```

[63]:
```
# adding a test student to test if the algorithm I wrote still works for␣
 ↪multiple grades... it should (since it showed above that it worked with␣
 ↪Taylor and Jordan's multiple school classes
# it's probably best to use a test student here so we don't contaminate the␣
 ↪data from Taylor and Jordon's tables
# this part of the question did not explicitly ask for a trigger- but we will␣
 ↪be implementing one for question 6 for student GPA
```

[64]:
```
%%sql

INSERT INTO Student VALUES (10, 'TestStudent', 'testEmail@example.com', 0);
INSERT INTO ClassGrade VALUES (10, 'CSPB3753', 4);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
```

[64]:
```
[]
```

[65]:
```
%%sql
SELECT
    s.sID,
    s.name,
    SUM(c.units * cg.grade) / SUM(c.units) AS computed_gpa
FROM Student AS s
JOIN ClassGrade AS cg ON s.sID = cg.sID
JOIN Class AS c ON cg.cID = c.cID
```

```
GROUP BY s.sID, s.name
HAVING s.sID = 10;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[65]:
```
[(10, 'TestStudent', Decimal('4.0000'))]
```

[66]:
```
%%sql
INSERT INTO ClassGrade VALUES (10, 'CSPB3308', 3);
INSERT INTO ClassGrade VALUES (10, 'CSPB4122', 2);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
```

[66]:
```
[]
```

[67]:
```
# then we test if the computed_gpa updated (which it should) based on new info␣
↪and grades slipped into the ClassGrade table
```

[68]:
```
%%sql
SELECT
    s.sID,
    s.name,
    SUM(c.units * cg.grade) / SUM(c.units) AS computed_gpa
FROM Student AS s
JOIN ClassGrade AS cg ON s.sID = cg.sID
JOIN Class AS c ON cg.cID = c.cID
GROUP BY s.sID, s.name
HAVING s.sID = 10;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[68]:
```
[(10, 'TestStudent', Decimal('3.1000'))]
```

[69]:
```
### END SOLUTION
```

4.2 Question 6 - Write a Trigger to update field of one table when data inserted into another table [10 pts]

Below you will write a Trigger to update field of one table when data inserted into another table. You will need to Write a Trigger to update the GPA for a student when a new grade is added to the ClassGrades table.

Create the trigger to update the Student's GPA every time a new ClassGrade is inserted.

[70]:
```
### BEGIN SOLUTION
```

[71]:
```
%%sql
DROP TRIGGER IF EXISTS update_student_gpa;

CREATE TRIGGER update_student_gpa
AFTER INSERT ON ClassGrade
FOR EACH ROW
BEGIN
    UPDATE Student AS s
    JOIN (
        SELECT
            cg.sID,
            SUM(c.units * cg.grade) / SUM(c.units) AS new_gpa
        FROM ClassGrade cg
        JOIN Class c ON c.cID = cg.cID
        WHERE cg.sID = NEW.sID
        GROUP BY cg.sID
    ) AS T ON s.sID = T.sID
    SET s.gpa = T.new_gpa;
END;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
0 rows affected.
```

[71]:
```
[]
```

[72]:
```
# I added DROP TRIGGER to drop any existing triggers that might be named␣
 ↪update_student_gpa due to me re-running the cell many times
# Then after making sure it doesn't exist, we create the trigger
# This trigger will fire after a new row is inserted into the ClassGrade table␣
 ↪and the trigger logic will only apply once per inserted row (key is that it␣
 ↪is not once per statement)
# BEGIN and END are to show the trigger body
# Then we update the student table rows, and inside JOIN is basically a␣
 ↪subquery that calculates new GPA for every single student that received a␣
 ↪new grade
# We take the logic per class, per unit, and what grade they received, and␣
 ↪update that with the GPA logic
# We only want to calculcate for the specific student whose new grade triggered␣
 ↪the insertion, and we alias the result as T, then update the GPA in s.gpa
```

[73]:
```
# show what triggers are so we have visibility
```

[74]:
```
%%sql

```

```
SHOW TRIGGERS;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[74]:
```
[('update_student_gpa', 'INSERT', 'ClassGrade', 'BEGIN\n    UPDATE Student AS
s\n    JOIN (\n        SELECT\n            cg.sID,\n            SUM(c.units *
cg.grade) / SUM(c.units) AS new_gpa\n      … (31 characters truncated) …
JOIN Class c ON c.cID = cg.cID\n        WHERE cg.sID = NEW.sID\n        GROUP BY
cg.sID\n    ) AS T ON s.sID = T.sID\n    SET s.gpa = T.new_gpa;\nEND', 'AFTER',
datetime.datetime(2025, 3, 6, 1, 7, 53, 710000), 'ONLY_FULL_GROUP_BY,STRICT_TRAN
S_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBST
ITUTION', 'bech1695@%', 'utf8mb3', 'utf8mb3_general_ci', 'utf8mb4_0900_ai_ci'),
 ('update_employee_count', 'INSERT', 'Employee', 'BEGIN\n  UPDATE Department SET
employee_count = employee_count + 1 \n    WHERE dID = new.dID;\nEND', 'AFTER',
datetime.datetime(2025, 3, 6, 1, 7, 53, 490000), 'ONLY_FULL_GROUP_BY,STRICT_TRAN
S_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBST
ITUTION', 'bech1695@%', 'utf8mb3', 'utf8mb3_general_ci', 'utf8mb4_0900_ai_ci')]
```

[75]:
```
### END SOLUTION
```

4.2.1 Test the Trigger by inserting more grades

You will need to create the trigger and in the trigger code run the calculation for the new GPA. Use that calculated value to then update the GPA field in the student's record.

You may need to add some more courses to allow you to insert more grades.

- Add a B+ (3.5) for Taylor in 3753
- Add a C for Jordon in 3287

[76]:
```
### BEGIN SOLUTION
```

[77]:
```
%%sql

INSERT INTO ClassGrade VALUES (1, 'CSPB3753', 3.5);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[77]:
```
[]
```

[78]:
```
%%sql

SELECT sID, name, gpa
FROM Student
WHERE sID IN (1, 2);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[78]:
```
[(1, 'Taylor', Decimal('3.3')), (2, 'Jordon', Decimal('0.0'))]
```

[79]:
```
# I wanted Jordon in the picture to make sure his GPA wasn't affected by the␣
 ↪long line of code I just wrote
```

[80]:
```
%%sql
SELECT sID, name, gpa
FROM Student
WHERE sID = 1;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[80]:
```
[(1, 'Taylor', Decimal('3.3'))]
```

[81]:
```
### END SOLUTION
```

[82]:
```
### BEGIN SOLUTION TESTING
```

[83]:
```
%%sql

INSERT INTO ClassGrade VALUES (2, 'CSPB3287', 2.0);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[83]:
```
[]
```

[84]:
```
%%sql

SELECT sID, name, gpa
FROM Student
WHERE sID IN (1, 2);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[84]:
```
[(1, 'Taylor', Decimal('3.3')), (2, 'Jordon', Decimal('3.0'))]
```

[85]:
```
# now Jordon's grades are also affected going from 3.42 -> 3.0
# Taylor's GPA went from 3.0 -> 3.3
```

[86]:
```
### END SOLUTION TESTING
```

4.3 Question 7 - Write a second Trigger to add a row to a table when an updated field of another table meets a criteria [10 pts]

Write a Trigger to add a notification to the Notifications table when a student's GPA falls below 2.5.

- To test your trigger, add an F (0) for Jordon in 4112 (GPA should drop to 2.3, which is below 2.5)

[87]:
```
### BEGIN SOLUTION 7.1
```

[88]:
```
# For 7.1, add notification, but 7.2 is about adding constraints so it's not␣
↪triggering all the time (like if the GPA
# is the same value, it will not adding a notification, 7.2 is a modification␣
↪of 7.1
```

[89]:
```
%%sql

DROP TRIGGER IF EXISTS notify_gpa_below_25;

CREATE TRIGGER notify_gpa_below_25
AFTER UPDATE ON Student
FOR EACH ROW
BEGIN
    -- Check if the new GPA is below 2.5
    IF NEW.gpa < 2.5 THEN
        INSERT INTO Notification (parentEmail, text)
        VALUES (
            NEW.parentEmail,
            CONCAT(
                'Dear Guardian, ', NEW.name,
                ' has just fallen below a 2.5 GPA. (Old GPA = ',
                OLD.gpa, ', New GPA = ', NEW.gpa, ')'
            )
        );
    END IF;
END;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
0 rows affected.
```

[89]:
```
[]
```

[90]:
```
%%sql

INSERT INTO ClassGrade VALUES (2, 'CSPB4122', 0.0);  -- Jordon gets an F in␣
↪CSPB4122
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[90]:
```
[]
```

[91]:
```
%%sql

SELECT *
FROM Notification;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[91]:
```
[('RileysDad@gmail.com', 'Riley needs to take some courses', 1),
 ('JordonsGram@gmail.com', 'Dear Guardian, Jordon has just fallen below a 2.5
GPA. (Old GPA = 3.0, New GPA = 2.3)', 0)]
```

[92]:
```
### END SOLUTION 7.1
```

Now, write a second trigger here that inserts a row in `Notification` with the parent's email and a message. The trigger should execute whenever a `Student` record is updated with a new GPA and that GPA is < 2.5. Make sure that you do not add another notification if the GPA does not change from the previous value.

A trigger like this can have a format similar to the following in SQLite:

```
create trigger XYZ
    after update of myColumn on myTable
    for each row when (condition in myTable)
    begin
        insert/update/delete etc.
     end
```

The text message for the notification should have informative text that includes student name, old GPA value, and new GPA value.
You may want to 14.8 String Functions and Operators in the MySQL documentation to see how to do string concatenation.

[93]:
```
### BEGIN SOLUTION 7.2
```

[94]:
```
# this solution shows that notifications don't always trigger
```

[95]:
```
%%sql

DROP TRIGGER IF EXISTS notify_gpa_below_25;

CREATE TRIGGER notify_gpa_below_25
AFTER UPDATE ON Student
FOR EACH ROW
```

```
BEGIN
    -- Check if GPA dropped below 2.5 and was previously 2.5 or above
    IF (NEW.gpa < 2.5) AND (OLD.gpa >= 2.5) THEN
        INSERT INTO Notification (parentEmail, text)
        VALUES (
            NEW.parentEmail,
            CONCAT('Dear Guardian, ', NEW.name, ' has just fallen below a 2.5␣
 ↪GPA. (Old GPA = ', OLD.gpa, ', New GPA = ', NEW.gpa, ').')
        );
    END IF;
END;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
0 rows affected.
0 rows affected.
```

[95]: []

[96]:
```
%%sql

DELETE FROM ClassGrade
WHERE sID = 2
  AND cID = 'CSPB4122';

INSERT INTO ClassGrade VALUES (2, 'CSPB4122', 0.0);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
```

[96]: []

[97]:
```
%%sql
SELECT * FROM Notification;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[97]:
```
[('RileysDad@gmail.com', 'Riley needs to take some courses', 1),
 ('JordonsGram@gmail.com', 'Dear Guardian, Jordon has just fallen below a 2.5
GPA. (Old GPA = 3.0, New GPA = 2.3)', 0)]
```

[98]:
```
### END SOLUTION 7.2
```

4.3.1 Now write some queries to test the trigger.

Make sure to test both the positive addition into Notifications and the negative addition when the GPA does not change.

If GPA was 3 and new course grade is 3, then there will be no change to the calculated GPA value and no notification should be added.

[99]:
```
### BEGIN SOLUTION
```

[100]:
```
%%sql

-- Before we begin, let's see what their GPAs and any existing notifications␣
 ↪look like

SELECT 'Current Students' AS Info, sID, name, gpa, parentEmail
FROM Student
WHERE name IN ('Taylor','Jordon');
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[100]:
```
[('Current Students', 1, 'Taylor', Decimal('3.3'), 'MorgansMom@gmail.com'),
 ('Current Students', 2, 'Jordon', Decimal('2.3'), 'JordonsGram@gmail.com')]
```

[101]:
```
%%sql
SELECT 'Current Notifications' AS Info, parentEmail, text
FROM Notification
WHERE parentEmail IN ('MorgansMom@gmail.com', 'JordonsGram@gmail.com');
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[101]:
```
[('Current Notifications', 'JordonsGram@gmail.com', 'Dear Guardian, Jordon has
just fallen below a 2.5 GPA. (Old GPA = 3.0, New GPA = 2.3)')]
```

[102]:
```
# only Jordon's GPA is below 2.5 so he's the one with existing notification
# let's test Taylor's GPA, seeing what adds a notification to the Notifications␣
 ↪table
```

[103]:
```
%%sql
-- Check Taylor's current GPA (sID = 1)

SELECT sID, name, gpa, parentEmail
FROM Student
WHERE sID = 1;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[103]:
```
[(1, 'Taylor', Decimal('3.3'), 'MorgansMom@gmail.com')]
```

[104]:
```
# let's now see what happens when we test for adding a new class that does not␣
↪influence Taylor's 3.3 gpa. We should not send a notification.
```

[105]:
```
%%sql
INSERT INTO Class (cID, name, units)
VALUES ('CSPB9999', 'Placeholder Class', 3);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[105]:
```
[]
```

[106]:
```
%%sql
INSERT INTO ClassGrade (sID, cID, grade)
VALUES (1, 'CSPB9999', 3.3);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[106]:
```
[]
```

[107]:
```
%%sql
SELECT sID, name, gpa, parentEmail
FROM Student
WHERE sID = 1;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[107]:
```
[(1, 'Taylor', Decimal('3.3'), 'MorgansMom@gmail.com')]
```

[108]:
```
# Taylor's GPA is still 3.3!
```

[109]:
```
%%sql
SELECT 'Current Notifications' AS Info, parentEmail, text
FROM Notification
WHERE parentEmail IN ('MorgansMom@gmail.com', 'JordonsGram@gmail.com');
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[109]:
```
[('Current Notifications', 'JordonsGram@gmail.com', 'Dear Guardian, Jordon has
just fallen below a 2.5 GPA. (Old GPA = 3.0, New GPA = 2.3)')]
```

[110]:
```
# only notification is still Jordon's!
# let's try what happens if we change the other courses to be lower/higher␣
↪while maintaining same GPA
```

[111]:
```
%%sql

-- Update Taylor's grades in two classes to the same (3.3) to see if trigger␣
 ↪fires
-- Adjust the cID values as needed

DELETE FROM ClassGrade
WHERE sID = 1 AND cID = 'CSPB3753';
INSERT INTO ClassGrade VALUES (1, 'CSPB3753', 3.6);

-- Originally CSPB3753 was 3.5 and CSPB3308 was 3.3 so I added 0.1 to one and␣
 ↪subtracted 0.1 to the other so the average would be the same to test if␣
 ↪notification would be sent

DELETE FROM ClassGrade
WHERE sID = 1 AND cID = 'CSPB3308';
INSERT INTO ClassGrade VALUES (1, 'CSPB3308', 3.2);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

[111]:
```
[]
```

[112]:
```
%%sql
-- Check Taylor's current record (sID = 1)

SELECT sID, name, gpa, parentEmail
FROM Student
WHERE sID = 1;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[112]:
```
[(1, 'Taylor', Decimal('3.3'), 'MorgansMom@gmail.com')]
```

[113]:
```
%%sql

SELECT *
FROM Notification
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
2 rows affected.
```

[113]:
```
[('RileysDad@gmail.com', 'Riley needs to take some courses', 1),
 ('JordonsGram@gmail.com', 'Dear Guardian, Jordon has just fallen below a 2.5
GPA. (Old GPA = 3.0, New GPA = 2.3)', 0)]
```

[114]:
```
# no notification would be sent, as expected because Taylor's grades changed,␣
 ↪but her GPA did not, and did not dip below 2.5
```

[115]:
```
%%sql

-- Update Taylor's grades in two classes to failing (0.0) so that her overall␣
 ↪GPA drops below 2.5
-- Adjust the cID values as needed

DELETE FROM ClassGrade
WHERE sID = 1 AND cID = 'CSPB3753';
INSERT INTO ClassGrade VALUES (1, 'CSPB3753', 0.0);


DELETE FROM ClassGrade
WHERE sID = 1 AND cID = 'CSPB3308';
INSERT INTO ClassGrade VALUES (1, 'CSPB3308', 0.0);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

[115]:
```
[]
```

[116]:
```
%%sql

-- Check Taylor's record again to confirm her GPA has dropped below 2.5.
SELECT sID, name, gpa, parentEmail
FROM Student
WHERE sID = 1;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[116]:
```
[(1, 'Taylor', Decimal('1.7'), 'MorgansMom@gmail.com')]
```

[117]:
```
%%sql

-- Check the Notification table for both Taylor and Jordon.
SELECT *
FROM Notification
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
3 rows affected.
```

[117]:
```
[('RileysDad@gmail.com', 'Riley needs to take some courses', 1),
 ('JordonsGram@gmail.com', 'Dear Guardian, Jordon has just fallen below a 2.5
GPA. (Old GPA = 3.0, New GPA = 2.3)', 0),
 ('MorgansMom@gmail.com', 'Dear Guardian, Taylor has just fallen below a 2.5
GPA. (Old GPA = 3.3, New GPA = 2.3).', 0)]
```

[118]:
```
# now her parents received notification
```

[119]:
```
%%sql

DELETE FROM ClassGrade
WHERE sID = 1 AND cID = 'CSPB3753';
INSERT INTO ClassGrade VALUES (1, 'CSPB3753', 3.5);


DELETE FROM ClassGrade
WHERE sID = 1 AND cID = 'CSPB3308';
INSERT INTO ClassGrade VALUES (1, 'CSPB3308', 3.3);
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

[119]:
```
[]
```

[120]:
```
# if we change back Taylor's grades, back to normal, it should update in the gpa
```

[121]:
```
%%sql

-- Check Taylor's GPA
SELECT sID, name, gpa, parentEmail
FROM Student
WHERE sID = 1;
```

```
 * mysql://bech1695:***@applied-sql.cs.colorado.edu:3306/bech1695
1 rows affected.
```

[121]:
```
[(1, 'Taylor', Decimal('3.3'), 'MorgansMom@gmail.com')]
```

[122]:
```
### END SOLUTION
```