REMARKABLE UNIVERSITY STUDENT GRADING SYSTEM

2809ICT Information & Security Management

OCTOBER 12, 2018

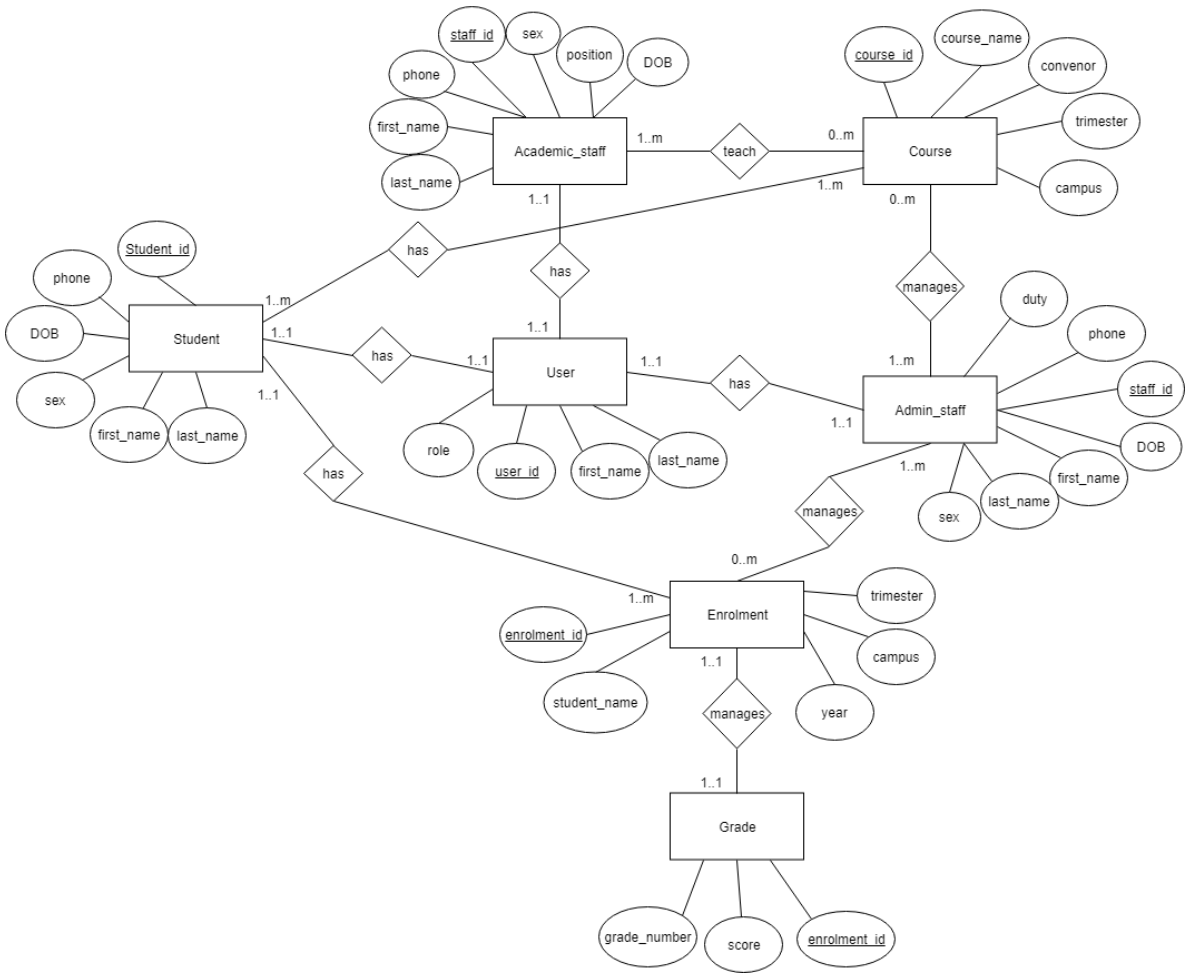
S5014818 MATTHEW HEATON
S5071539 DAVID SAELE

Database Implementation

Database Design

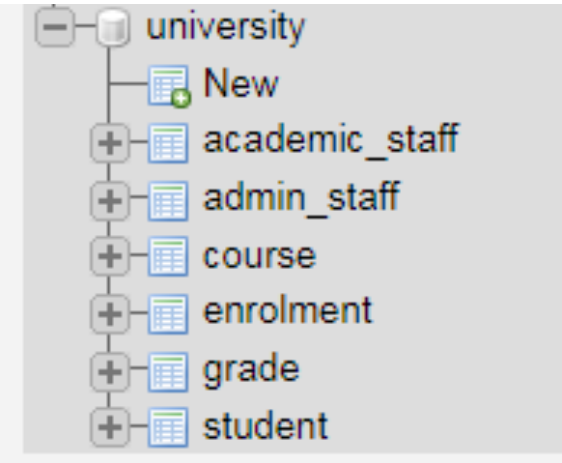
Assumptions:

- Academic staff may teach 0, 1 or many courses.
- A course must have one academic staff, but can also have more than one.
- A student must have at least one course to be a student, can also have many courses.
- A course must have at least one student in it, can also have many students.
- A course must be managed by at least one admin staff and can be managed by multiple.
- An admin staff can manage 0, 1 or many courses.
- Students and all staff have only one user.
- Admin staff manages 0 or more enrollments, while enrollments must be managed by 1 or more staff.
- Students can have 1 or many enrollments, but an enrollment can only have one student.
- Each enrollment has one grade and each grade has one enrollment.
- Added sex, date of birth and phone attribute for academic and admin staff.
- Added student_ID for the grade table to identify which students holds which grade.



SQL Implementation

Added all the tables necessary in the university database such as academic_staff, admin_staff, course, enrollment, grade and student.



Provided below is the academic_staff table with staff_ID, first names, last names, positions and the addition of date of birth, sex and phone numbers.

```
SELECT *
FROM `academic_staff`
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Re

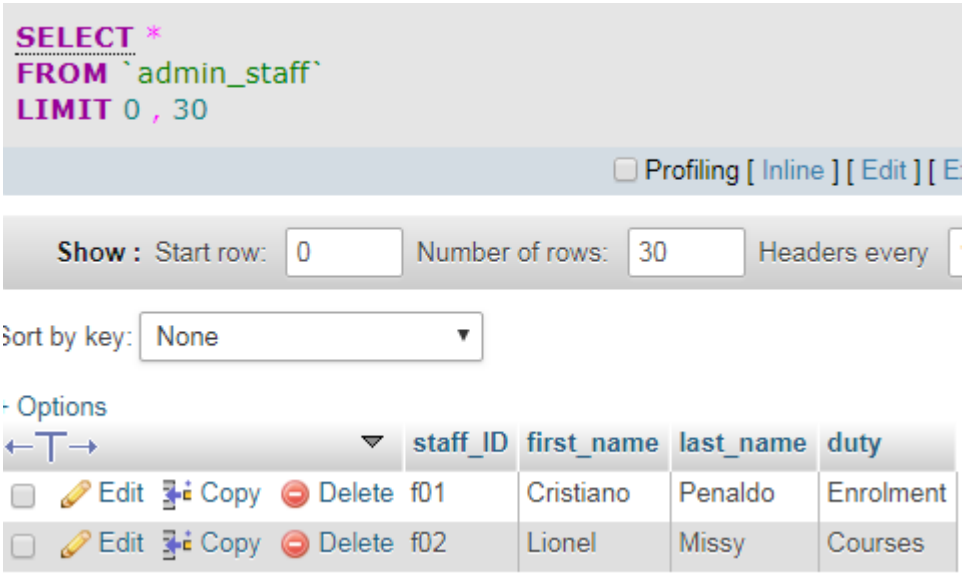
Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

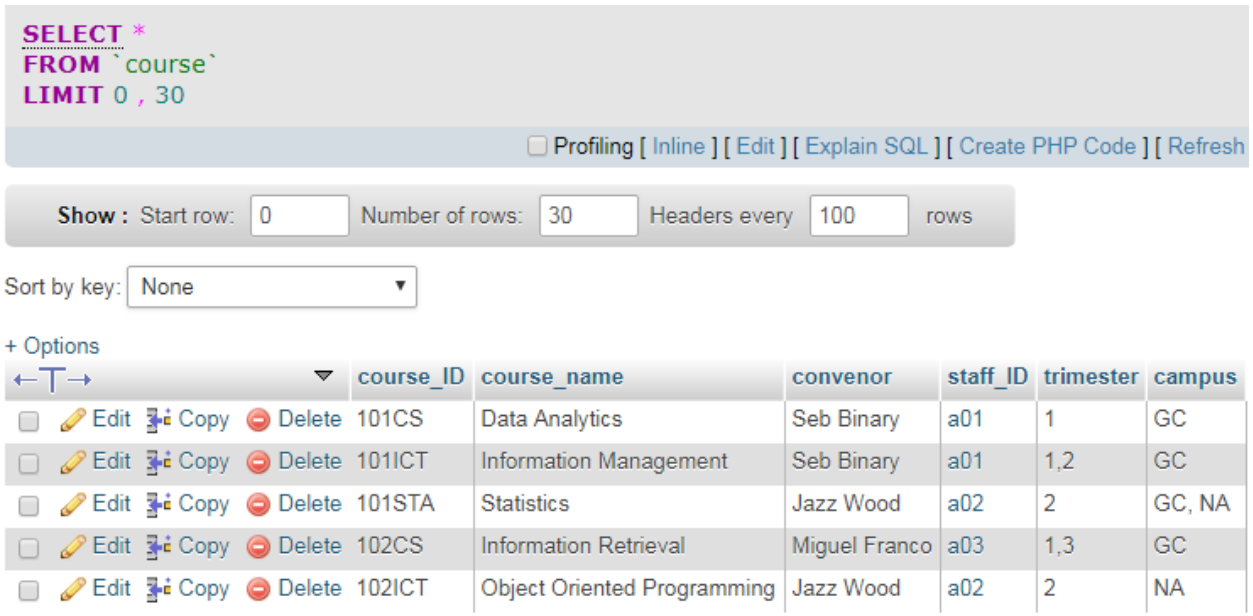
+ Options

	staff_ID	first_name	last_name	position	dob	sex	phone
Edit Copy Delete	a01	Seb	Binary	Professor	1986-09-12	M	0755527076
Edit Copy Delete	a02	Jazz	Wood	Asso. Professor	1976-10-10	M	0754427099
Edit Copy Delete	a03	Miguel	Franco	Lecturer	1990-05-12	M	0752270933

Below is the adim_staff table with staff_ID, first names, last names and duties.



In the image below the course table is provided including course_ID, course names, convenor names, staff_ID, the trimester the course in available and which campus the course is available.



Below is the enrollment table with enrollment_ID, course_ID, student_ID, students full name, year they enrolled, trimester they enrolled and which campus they enrolled.

	enrolment_ID	course_ID	student_ID	student_name	year	trimester	campus
Edit Copy Delete	e01	101ICT	s01	Angela Merkal	2017	1	GC
Edit Copy Delete	e02	101STA	s01	Angela Merkal	2017	1	GC
Edit Copy Delete	e03	101CS	s01	Angela Merkal	2017	1	GC
Edit Copy Delete	e04	102ICT	s02	Donaldo True	2018	2	NA
Edit Copy Delete	e05	101CS	s02	Donaldo True	2017	1	GC
Edit Copy Delete	e06	102CS	s03	Hillarious Blinton	2018	3	GC
Edit Copy Delete	e07	102ICT	s04	Tarra Obana	2018	2	NA
Edit Copy Delete	e08	101STA	s04	Tarra Obana	2017	1	NA
Edit Copy Delete	e09	101CS	s04	Tarra Obana	2017	1	GC
Edit Copy Delete	e10	102CS	s04	Tarra Obana	2018	1	GC

This is the grade table with enrolment_ID, score students are given, grade_number and the additional student_ID to know which student has what grade.

+ Options

	enrolment_ID	student_ID	score	grade_number
Edit Copy Delete	e01	s01	75	6
Edit Copy Delete	e02	s01	80	6
Edit Copy Delete	e03	s01	92	7
Edit Copy Delete	e04	s02	86	7
Edit Copy Delete	e05	s02	71	5
Edit Copy Delete	e06	s03	65	5
Edit Copy Delete	e07	s04	55	4
Edit Copy Delete	e08	s04	80	6
Edit Copy Delete	e09	s04	80	6
Edit Copy Delete	e10	s04	86	7

Provided below is the student table with student_ID, first names, last names, date of birth, sex and phone numbers.

```
SELECT *
FROM `student`
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [Crea

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

	student_ID	first_name	last_name	dob	sex	phone
Edit Copy Delete	s01	Angela	Merkal	1991-01-01	F	543210
Edit Copy Delete	s02	Donaldo	True	1992-02-02	M	123456
Edit Copy Delete	s03	Hillarious	Blinton	1993-03-03	F	112233
Edit Copy Delete	s04	Tarra	Obana	1994-04-04	M	221134

For the SQL code please look at the UniversityDB_Assignment-2.sql file provided.

Users and Privileges

Assign Table-level Privileges

As a general rule, all users should have restricted access to the database according to their roles. The admin staff users were only granted access according to what their job included. For example, the user Cristiano was granted all privileges only on the enrolment table, while Lionel were granted all privileges on the course table. They were also given with grant option, so they can further assign those same privileges to new admin staff in the future.

For students and academic staff, they have to see information about each course in order to be able to choose the courses. However, they would only need access to read the information, so they will only be granted a select on the course table in the university database. In that way, they are only able to read information about the courses but cannot update or delete the files.

It is useful for academic staff to see the names and genders for each student, so in order to be able to do that, a view was created which displays those attributes. This view was granted to every academic staff. The codes for granting these privileges is provided below.

```
CREATE VIEW names_gender AS SELECT first_name, last_name, sex
FROM university.student;
```

```
GRANT ALL PRIVILEGES ON university.enrolment TO 'Cristiano'@'%' WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON university.course TO 'Lionel'@'%' WITH GRANT OPTION;
GRANT SELECT ON university.course TO 'Angela'@'%', 'Donaldo'@'%', 'Hillarious'@'%',
'Tarra'@'%', 'Seb'@'%', 'Jazz'@'%', 'Miguel'@'%';
GRANT SELECT ON university.names_gender TO 'Seb'@'%', 'Jazz'@'%', 'Miguel'@'%';
```

Create Views and Related Privileges

To make it possible for students to view their grades, a view was created for each student with a where clause specifying that they only have access to their own grades. The where clause uses the attribute student_id, which makes sure that when the user is logged in, he/she can only see the grades connected to his/her user. After the view was created, each user was granted a select access which only lets the user read the grade, and not modify them.

```
CREATE VIEW angela_grades AS SELECT student_ID, score, grade_number
FROM university.grade WHERE student_ID = 's01';

CREATE VIEW donaldo_grades AS SELECT student_ID, score, grade_number
FROM university.grade WHERE student_ID = 's02';

CREATE VIEW hillarious_grades AS SELECT student_ID, score, grade_number
FROM university.grade WHERE student_ID = 's03';

CREATE VIEW tarra_grades AS SELECT student_ID, score, grade_number
FROM university.grade WHERE student_ID = 's04';

GRANT SELECT ON university.angela_grades TO 'Angela'@'%';
GRANT SELECT ON university.donaldo_grades TO 'Donaldo'@'%';
GRANT SELECT ON university.hillarious_grades TO 'Hillarious'@'%';
GRANT SELECT ON university.tarra_grades TO 'Tarra'@'%';
```

For the academic staff to have an overview of how many students are enrolled in their course, it is handy for them to have a read access to the enrolment table. To allow that, a view for every academic staff was created where the enrolment table and the course table were joined. The reason for the join is because we have to specify that each staff only have access to their own courses by specifying that the staff_id connects to the correct user. Since there are no staff_id column in the enrolment table, the join is created to the course table make sure that each user only has access to their own course. Each view is granted to the appropriate user, with a read-only permission.

```
CREATE VIEW seb_course AS SELECT c.course_ID, e.enrolment_ID, e.student_ID,
e.student_name, e.year, e.trimester, e.campus FROM university.course c, university.enrolment e
WHERE c.course_ID = e.course_ID AND c.staff_ID = 'a01';

CREATE VIEW jazz_course AS SELECT c.course_ID, e.enrolment_ID, e.student_ID,
e.student_name, e.year, e.trimester, e.campus FROM university.course c, university.enrolment e
WHERE c.course_ID = e.course_ID AND c.staff_ID = 'a02';

CREATE VIEW miguel_course AS SELECT c.course_ID, e.enrolment_ID, e.student_ID, e.student_name,
e.year, e.trimester, e.campus FROM university.course c, university.enrolment e
WHERE c.course_ID = e.course_ID AND c.staff_ID = 'a03';

GRANT SELECT ON university.seb_course TO 'Seb'@'%';
GRANT SELECT ON university.jazz_course TO 'Jazz'@'%';
GRANT SELECT ON university.miguel_course TO 'Miguel'@'%';
```

Academic staff must be able to modify and insert the grades for every student throughout the trimester. To grant that permission, a view for every staff was created which displays the student_ID, name, year, trimester, campus, score and grade for the students enrolled in their courses. For the staff to be able to modify the grades, every staff was granted select and update access on their appropriate view. However, it is not necessary for the staff to be able to modify the other attributes except for score and grades, so they are only granted update on those columns.

```
CREATE VIEW seb_modify_grades AS SELECT e.course_ID, e.student_ID, e.student_name,
e.year, e.trimester, e.campus,  g.score, g.grade_number
FROM university.grade g, university.enrolment e, university.course c
WHERE g.enrolment_ID = e.enrolment_ID AND e.course_ID = c.course_ID AND c.staff_ID = 'a01';

CREATE VIEW jazz_modify_grades AS SELECT e.course_ID, e.student_ID, e.student_name,
e.year, e.trimester, e.campus,  g.score, g.grade_number
FROM university.grade g, university.enrolment e, university.course c
WHERE g.enrolment_ID = e.enrolment_ID AND e.course_ID = c.course_ID AND c.staff_ID = 'a02';

CREATE VIEW miguel_modify_grades AS SELECT e.course_ID, e.student_ID, e.student_name,
e.year, e.trimester, e.campus,  g.score, g.grade_number
FROM university.grade g, university.enrolment e, university.course c
WHERE g.enrolment_ID = e.enrolment_ID AND e.course_ID = c.course_ID AND c.staff_ID = 'a03';

GRANT SELECT, UPDATE (score, grade_number) ON university.seb_modify_grades TO 'Seb'@'%';
GRANT SELECT, UPDATE (score, grade_number) ON university.jazz_modify_grades TO 'Jazz'@'%';
GRANT SELECT, UPDATE (score, grade_number) ON university.miguel_modify_grades TO 'Miguel'@'%';
```

Access Matrix

	Grades	Course	Enrolment	Student
Cristiano			Read Write	

Lionel		Read Write		
Seb	Read Write	Read	Read (only for own courses)	Read (only names and gender)
Jazz	Read Write	Read	Read (only for own courses)	Read (only names and gender)
Miguel	Read Write	Read	Read (only for own courses)	Read (only names and gender)
Angela	Read (only for own grades)	Read		
Donaldo	Read (only for own grades)	Read		
Hillarious	Read (only for own grades)	Read		
Tarra	Read (only for own grades)	Read		

SQL Injection

To inject the database, we entered malicious code into the input forms to modify the database. The modification we tried to do was to delete the grade table entirely from the database. To do so, we gave the input form a sample student called Example Name, with student number s06, birthday 2000-01-01 with 010101 as phone number and male as gender. In the last input field, we added ‘); to close the input bracket in the code and then gave the input DROP TABLE grade; to delete the grade table. As you can see in the image below, the record was created successfully, and the table grade was dropped.

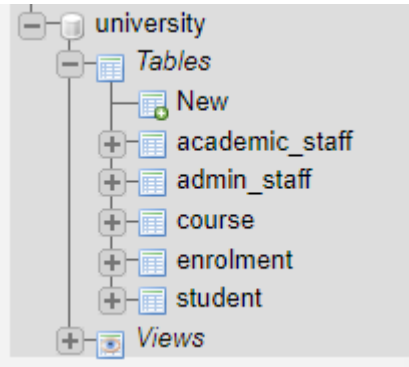
Connected successfully

New record created successfully

Input 1 s06
Input 2 Example
Input 3 Name
Input 4 2000-01-01
Input 5 M
Input 6 010101'); DROP TABLE grade;

Return to the form

When we tried to access the table grade, it was been deleted from the database as expected.



When we looked at the student table, the sample student was created as expected.

	student_ID	first_name	last_name	dob	sex	phone
Edit Copy Delete	s01	Angela	Merkal	1991-01-01	F	543210
Edit Copy Delete	s02	Donaldo	True	1992-02-02	M	123456
Edit Copy Delete	s03	Hillarious	Blinton	1993-03-03	F	112233
Edit Copy Delete	s04	Tarra	Obana	1994-04-04	M	221134
Edit Copy Delete	s06	Example	Name	2000-01-01	M	10101

Database Backup Strategy

A backup strategy is important as there is a very real chance of confidentiality, integrity and availability failures. For instance, an attacker may find a way to drop certain tables giving less availability in the database. A good example is if a attacker drops student grades, teachers

would have a hard time figuring out what the student deserves score wise. Also the student would not know exactly what grade they would get and if it would be the same as it was before the attack.

An attacker could also target confidential data such as phone numbers. A good example here is the student table of which hold student phone numbers. So a attacker could use them for a number of schemes for example telemarketing. Integrity could be affected by an attacker such as changing a name of a student. In our database for example if you changed a student's name it would be changed everywhere. You need to have a backup strategy as confidentiality, integrity and availability can be also affected by bugs in code. Therefore we suggest a cloud-based storage service to have regular backups.

The database overall should be backed up but there are parts on the database that should be backed up more than others. Overall, the database should be backed up at least every week as most of the data will need to be changed often therefore not a lot of risk. The grades should be backed up every day because student attackers will want to change their grades but also they are more prone to change then all other data.

For the overall database backup we believe we should retain 5 backups before the newest backup overwrites the oldest. It makes sense to keep record of 5 weeks of data changes in a database overall as it may take at least 2 to 3 weeks for a user to notice a data change. For example if a students name changes from Nathan to James, Nathan may not realise because why would he need to looking at his name. As we talked about before we believe grades should be retained everyday therefore 7 times per week before the newest backup overwrites the oldest. This is because of the many times grades may need to change in a week in assessment weeks.

SQL RECOVERY

It is important with a backup file to recover the database if it were to be deleted or altered with by unauthorised people. To do so, the command in the figure below was used to create the backup file called university_backup.sql.

```
cabox@box-codeanywhere:~/workspace$ mysqldump --user=root university > university_backup.sql
cabox@box-codeanywhere:~/workspace$ ls -l
total 16
-rw-r--r-- 1 cabox cabox  998 Oct 11 01:43 index.html
-rw-r--r-- 1 cabox cabox 1197 Oct 11 01:43 main.php
-rw-r--r-- 1 cabox cabox 7834 Oct 11 04:33 university_backup.sql
```

Then the database was backed up by using the mysqldump -u root -p university > university_backup.sql. The UNIX prompt then asked for a password, but there is no password

for root, so we just pressed enter and the university database was backed up. To connect to MySQL the command mysql --user=root university was used.

```
cabox@box-codeanywhere:~/workspace$ mysqldump -u root -p university > university_backup.sql
Enter password:
cabox@box-codeanywhere:~/workspace$ mysql --user=root university
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 63
Server version: 5.5.38-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

After entering MySQL, we dropped the entire database, which deleted everything that was stored in it. In the image below, we tried to show tables after deleting the database, but got an error message stating that no database selected because there was no database there. Then we created a new database called university. When the new university database was created, we used the backup file to populate the database. The command that was used was source university_backup.sql.

```
mysql> drop database university;
Query OK, 6 rows affected (0.09 sec)

mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> create database if not exists university;
Query OK, 1 row affected (0.01 sec)

mysql> use university;
Database changed
mysql> source university_backup.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

After the database was populated, we entered the command show tables to see if it worked, which it did. As the picture below illustrates, all the tables are at place, and as an example, the student and enrolment table are populated as expected.

```
mysql> show tables;
+----------------------+
| Tables_in_university |
+----------------------+
| academic_staff       |
| admin_staff          |
| course               |
| enrolment            |
| grade                |
| student              |
+----------------------+
6 rows in set (0.00 sec)

mysql> select * from student;
+------------+------------+-----------+------------+-----+--------+
| student_ID | first_name | last_name | dob        | sex | phone  |
+------------+------------+-----------+------------+-----+--------+
| s01        | Angela     | Merkal    | 1991-01-01 | F   | 543210 |
| s02        | Donaldo    | True      | 1992-02-02 | M   | 123456 |
| s03        | Hillarious | Blinton   | 1993-03-03 | F   | 112233 |
| s04        | Tarra      | Obana     | 1994-04-04 | M   | 221134 |
+------------+------------+-----------+------------+-----+--------+
4 rows in set (0.00 sec)

mysql>
```

```
mysql> select * fromenrolment;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to yo
L server version for the right syntax to use near 'fromenrolment' at line 1
mysql> select * from enrolment;
+--------------+-----------+------------+--------------------+------+-----------+--------+
| enrolment_ID | course_ID | student_ID | student_name       | year | trimester | campus |
+--------------+-----------+------------+--------------------+------+-----------+--------+
| e01          | 101ICT    | s01        | Angela Merkal      | 2017 |         1 | GC     |
| e02          | 101STA    | s01        | Angela Merkal      | 2017 |         1 | GC     |
| e03          | 101CS     | s01        | Angela Merkal      | 2017 |         1 | GC     |
| e04          | 102ICT    | s02        | Donaldo True       | 2018 |         2 | NA     |
| e05          | 101CS     | s02        | Donaldo True       | 2017 |         1 | GC     |
| e06          | 102CS     | s03        | Hillarious Blinton | 2018 |         3 | GC     |
| e07          | 102ICT    | s04        | Tarra Obana        | 2018 |         2 | NA     |
| e08          | 101STA    | s04        | Tarra Obana        | 2017 |         1 | NA     |
| e09          | 101CS     | s04        | Tarra Obana        | 2017 |         1 | GC     |
| e10          | 102CS     | s04        | Tarra Obana        | 2018 |         1 | GC     |
+--------------+-----------+------------+--------------------+------+-----------+--------+
10 rows in set (0.00 sec)

mysql>
```