(/)

Machine Learning Pipeline
Average: **100.0%**

Databases

100%

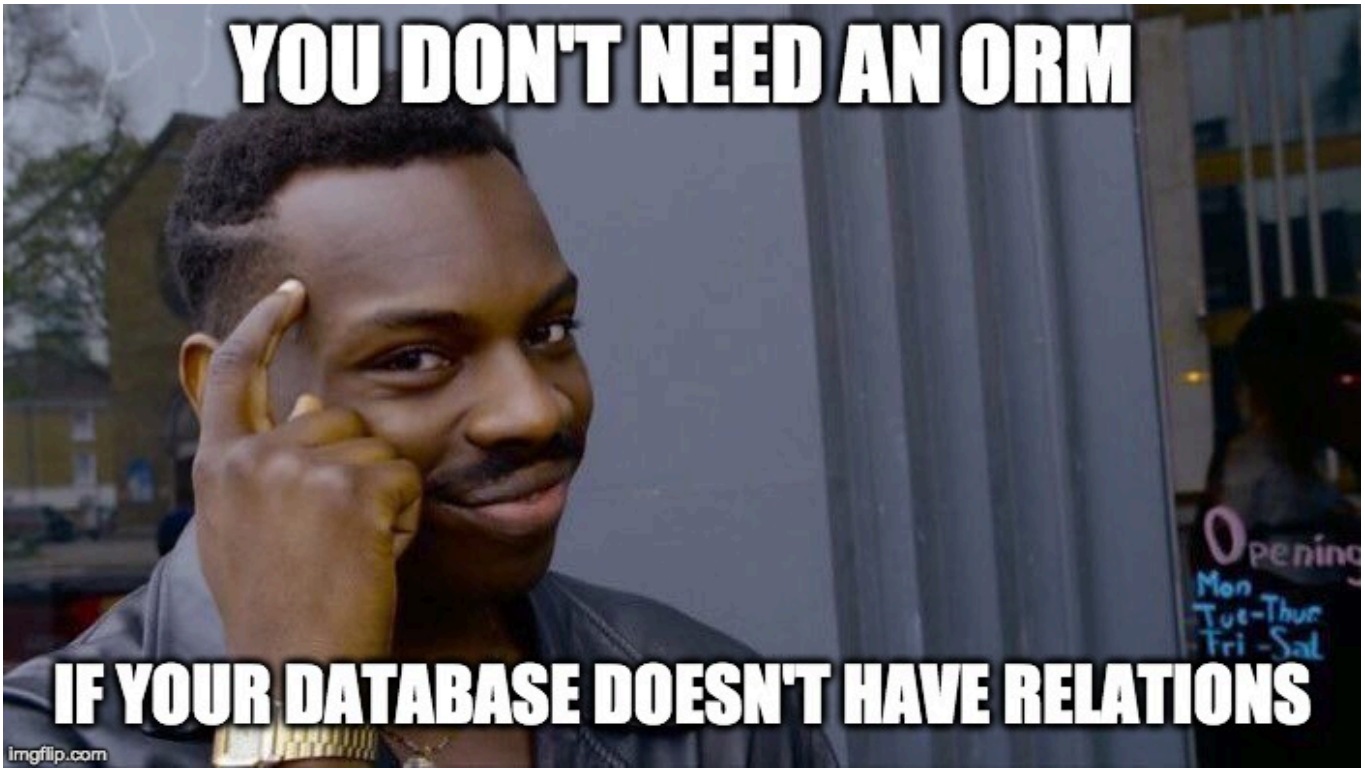
Amateur

By: Alexa Orrico, Software Engineer at Holberton School

Weight: 1

Your score will be updated as you progress.

Project will start Oct 27, 2024 12:00 AM, must end by Nov 2, 2024 11:59 PM



After fetching data via APIs, storing them is also really important for training a Machine Learning model.

You have multiple option:

- Relation database

- Not Relation database
- Key-Value storage
- Document storage
- Data Lake
- etc.

In this project, you will touch the first 2: relation and not relation database.

Relation databases are mainly used for application, not for source of data for training your ML models, but it can be really useful for the data processing, labeling and injection in another data storage. In this project, you will play with basic SQL commands but also create automation and computing on your data directly in SQL - less load at your application level since the computing power is dispatched to the database.

Not relation databases, known as NoSQL, will give you flexibility on your data: document, versioning, not a fix schema, no validation to improve performance, complex lookup, etc.

Resources

Read or watch:

- MySQL:
  - What is Database & SQL? (/rltoken/ZN5JtMadhRA6t09A-AxzIg)
  - MySQL Cheat Sheet (/rltoken/-D4RqM33usRh_5AYZZ7T9w)
  - MySQL 5.7 SQL Statement Syntax (/rltoken/EKbXbz-7stlQXRBoy6q6GQ)
  - MySQL Performance: How To Leverage MySQL Database Indexing (/rltoken/Us3U3hSMlY7xi192P4cMwQ)
  - Stored Procedure (/rltoken/7X-_WvYe7lLjDMTcL5etow)
  - Triggers (/rltoken/7fCKaiCol85EXfhclyrEHw)
  - Views (/rltoken/pxJJ0c-KEmBAsOwQlJcoaA)
  - Functions and Operators (/rltoken/PMLdYuP1SqdVxj4FJWWQaQ)
  - Trigger Syntax and Examples (/rltoken/-myQiKQqN61tUhNpgXNPFw)
  - CREATE TABLE Statement (/rltoken/ocfsv2hcJrRL8pt2DAxUHQ)
  - CREATE PROCEDURE and CREATE FUNCTION Statements (/rltoken/z8KDOy6ulogKWpTHc91PqA)
  - CREATE INDEX Statement (/rltoken/1O4zViWK1nzZ7Ymh74wEEw)
  - CREATE VIEW Statement (/rltoken/RQFGt2t0m_aXVxFHE7K7cA)
- NoSQL:
  - NoSQL Databases Explained (/rltoken/VApjv1JgTRwGicviwTEwow)
  - What is NoSQL ? (/rltoken/HJXCiEUMLQ4d6eTvPCnvZQ)
  - Building Your First Application: An Introduction to MongoDB (/rltoken/xRb2vM-tlqKg_DtDsSnSng)
  - MongoDB Tutorial 2 : Insert, Update, Remove, Query (/rltoken/mCaX1q0RhqCmTkzt42N4qg)
  - Aggregation (/rltoken/GQFRV_7ooqOFOqHlDi2edw)
  - Introduction to MongoDB and Python (/rltoken/6tROcxei7jXCAIaconbpjQ)
  - mongo Shell Methods (/rltoken/H5_nXBS_70vQU4llBeKYTw)
  - The mongo Shell (/rltoken/iJzL81AsPIydPPTYIaByUA)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/PgWMfK5aZNE2kgsOxpUTmQ), **without the help of Google**:

General

- What's a relational database
- What's a none relational database
- What is difference between SQL and NoSQL
- How to create tables with constraints
- How to optimize queries by adding indexes
- What is and how to implement stored procedures and functions in MySQL
- What is and how to implement views in MySQL
- What is and how to implement triggers in MySQL
- What is ACID
- What is a document storage
- What are NoSQL types
- What are benefits of a NoSQL database
- How to query information from a NoSQL database
- How to insert/update/delete information from a NoSQL database
- How to use MongoDB

Requirements

General

- A `README.md` file, at the root of the folder of the project, is mandatory
- All your SQL files will be executed on Ubuntu 16.04 LTS (or 18.04) using `MySQL 5.7` (version 5.7.30)
- All your SQL queries should have a comment just before (i.e. syntax above)
- All SQL keywords should be in uppercase (`SELECT`, `WHERE` ...)
- All your Mongo files will be interpreted/compiled on Ubuntu 16.04 LTS (or 18.04) using `MongoDB` (version 4.2)
- The first line of all your Mongo files should be a comment: `// my comment`
- All your Python files will be interpreted/compiled on Ubuntu 16.04 LTS (or 18.04) using `python3` (version 3.5 or 3.7) and `PyMongo` (version 3.10)
- The first line of all Python your files should be exactly `#!/usr/bin/env python3`
- Your Python code should use the `pycodestyle` style (version 2.5.*)
- All your Python modules should have a documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)
- All your Python functions should have a documentation (`python3 -c 'print(__import__("my_module").my_function.__doc__)'`
- Your Python code should not be executed when imported (by using `if __name__ == "__main__"` :)
- All your files should end with a new line
- The length of your files will be tested using `wc`

More Info

MySQL

Comments for your SQL file:

```
$ cat my_script.sql
-- 3 first students in the Batch ID=3
-- because Batch 3 is the best!
SELECT id, name FROM students WHERE batch_id = 3 ORDER BY created_at DESC LIMIT 3;
$
```

Install locally

```
$  sudo apt-get install mysql-server
...
$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.31-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2020, 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.

mysql>
```

Use “container-on-demand” to run MySQL

- Ask for container `Ubuntu 18.04 - Python 3.7`
- Connect via SSH
- Or via the WebTerminal
- In the container, you should start MySQL before playing with it:

```
$ service mysql start
 * MySQL Community Server 5.7.30 is started
$
$ cat 0-list_databases.sql | mysql -uroot -p my_database
Enter password:
Database
information_schema
mysql
performance_schema
sys
$
```

In the container, credentials are `root/root`

How to import a SQL dump

(/)

```
$ echo "CREATE DATABASE hbtn_0d_tvshows;" | mysql -uroot -p
Enter password:
$ curl "https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_progra
mming+/274/hbtn_0d_tvshows.sql" -s | mysql -uroot -p hbtn_0d_tvshows
Enter password:
$ echo "SELECT * FROM tv_genres" | mysql -uroot -p hbtn_0d_tvshows
Enter password:
id  name
1   Drama
2   Mystery
3   Adventure
4   Fantasy
5   Comedy
6   Crime
7   Suspense
8   Thriller
$
```

MongoDB

Install MongoDB 4.2

Official installation guide (/rltoken/Hwj2_7ba_gM7jrnjQ1kUlA)

```
$ wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | apt-key add -
$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2
multiverse" > /etc/apt/sources.list.d/mongodb-org-4.2.list
$ sudo apt-get update
$ sudo apt-get install -y mongodb-org
...
$  sudo service mongod status
mongod start/running, process 3627
$ mongo --version
MongoDB shell version v4.2.8
git version: 43d25964249164d76d5e04dd6cf38f6111e21f5f
OpenSSL version: OpenSSL 1.1.1  11 Sep 2018
allocator: tcmalloc
modules: none
build environment:
    distmod: ubuntu1804
    distarch: x86_64
    target_arch: x86_64
$
$ pip3 install pymongo
$ python3
>>> import pymongo
>>> pymongo.__version__
'3.10.1'
```

Potential issue if documents creation doesn't work or this error: `Data directory /data/db not found., terminating` (source (/rltoken/MafYCQ_i6lVz6qnVtwVvoQ) and source (/rltoken/iOtfFs5GTDI6wowSp9_CuA))

```
$ sudo mkdir -p /data/db
```

Use "container-on-demand" to run MongoDB

- Ask for container `Ubuntu 18.04 - MongoDB`
- Connect via SSH
- Or via the WebTerminal
- In the container, you should start MongoDB before playing with it:

```
$ service mongod start
* Starting database mongod                                              [ OK ]
$
$ cat 0-list_databases | mongo
MongoDB shell version v4.2.8
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("70f14b38-6d0b-48e1-a9a4-0534bcf15301") }
MongoDB server version: 4.2.8
admin   0.000GB
config  0.000GB
local   0.000GB
bye
$
```

Tasks

0. Create a database

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that creates the database `db_0` in your MySQL server.

- If the database `db_0` already exists, your script should not fail
- You are not allowed to use the `SELECT` or `SHOW` statements

```
guillaume@ubuntu:~/$ cat 0-create_database_if_missing.sql | mysql -hlocalhost -uroot -p
Enter password:
guillaume@ubuntu:~/$ echo "SHOW databases;" | mysql -hlocalhost -uroot -p
Enter password:
Database
information_schema
db_0
mysql
performance_schema
guillaume@ubuntu:~/$ cat 0-create_database_if_missing.sql | mysql -hlocalhost -uroot -p
Enter password:
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `0-create_database_if_missing.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

1. First table

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that creates a table called `first_table` in the current database in your MySQL server.

- `first_table` description:
  - `id` INT
  - `name` VARCHAR(256)
- The database name will be passed as an argument of the `mysql` command
- If the table `first_table` already exists, your script should not fail
- You are not allowed to use the `SELECT` or `SHOW` statements

```
guillaume@ubuntu:~/$ cat 1-first_table.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
guillaume@ubuntu:~/$ echo "SHOW TABLES;" | mysql -hlocalhost -uroot -p db_0
Enter password:
Tables_in_db_0
first_table
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`

- File: `1-first_table.sql`

(/)

Review your work | >_ Get a sandbox **6/6** pts

2. List all in table

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that lists all rows of the table `first_table` in your MySQL server.

- All fields should be printed
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat 2-list_values.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `2-list_values.sql`

Review your work | >_ Get a sandbox **6/6** pts

3. First add

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that inserts a new row in the table `first_table` in your MySQL server.

- New row:
  - `id = 89`
  - `name = Holberton School`
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat 3-insert_value.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
guillaume@ubuntu:~/$ cat 2-list_values.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
id  name
89  Holberton School
guillaume@ubuntu:~/$ cat 3-insert_value.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
guillaume@ubuntu:~/$ cat 3-insert_value.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
guillaume@ubuntu:~/$ cat 2-list_values.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
id  name
89  Holberton School
89  Holberton School
89  Holberton School
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `3-insert_value.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

4. Select the best

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that lists all records with a `score >= 10` in the table `second_table` in your MySQL server.

- Results should display both the score and the name (in this order)
- Records should be ordered by score (top first)
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat setup.sql
-- Create table and insert data
CREATE TABLE IF NOT EXISTS second_table (
    id INT,
    name VARCHAR(256),
    score INT
);
INSERT INTO second_table (id, name, score) VALUES (1, "Bob", 14);
INSERT INTO second_table (id, name, score) VALUES (2, "Roy", 3);
INSERT INTO second_table (id, name, score) VALUES (3, "John", 10);
INSERT INTO second_table (id, name, score) VALUES (4, "Bryan", 8);

guillaume@ubuntu:~/$ cat setup.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
guillaume@ubuntu:~/$ cat 4-best_score.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
score   name
14  Bob
10  John
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `4-best_score.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

5. Average

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that computes the score average of all records in the table `second_table` in your MySQL server.

- The result column name should be `average`
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat setup.sql
-- Create table and insert data
CREATE TABLE IF NOT EXISTS second_table (
    id INT,
    name VARCHAR(256),
    score INT
);
INSERT INTO second_table (id, name, score) VALUES (1, "Bob", 14);
INSERT INTO second_table (id, name, score) VALUES (2, "Roy", 5);
INSERT INTO second_table (id, name, score) VALUES (3, "John", 10);
INSERT INTO second_table (id, name, score) VALUES (4, "Bryan", 8);

guillaume@ubuntu:~/$ cat setup.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
guillaume@ubuntu:~/$ cat 5-average.sql | mysql -hlocalhost -uroot -p db_0
Enter password:
average
9.25
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `5-average.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

6. Temperatures #0

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Import in `hbtn_0c_0` database this table dump: download (https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_programming+/272/temperatures.sql)

Write a script that displays the average temperature (Fahrenheit) by city ordered by temperature (descending).

```
guillaume@ubuntu:~/$ cat 6-avg_temperatures.sql | mysql -hlocalhost -uroot -p hbtn_0c_0
Enter password:
city    avg_temp
Chandler    72.8627
Gilbert 71.8088
Pismo beach 71.5147
San Francisco   71.4804
Sedona  70.7696
Phoenix 70.5882
Oakland 70.5637
Sunnyvale   70.5245
Chicago 70.4461
San Diego   70.1373
Glendale    70.1225
Sonoma  70.0392
Yuma    69.3873
San Jose    69.2990
Tucson  69.0245
Joliet  68.6716
Naperville  68.1029
Tempe   67.0441
Peoria  66.5392
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `6-avg_temperatures.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

7. Temperatures #2

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Import in `hbtn_0c_0` database this table dump: download (https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_programming+/272/temperatures.sql) (same as `Temperatures #0`)

Write a script that displays the max temperature of each state (ordered by State name).

```
guillaume@ubuntu:~/$ cat 7-max_state.sql | mysql -hlocalhost -uroot -p hbtn_0c_0
Enter password:
state   max_temp
AZ  110
CA  110
IL  110
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `7-max_state.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

8. Genre ID by show

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Import the database dump from `hbtn_0d_tvshows` to your MySQL server: download (https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_programming+/274/hbtn_0d_tvshows.sql)

Write a script that lists all shows contained in `hbtn_0d_tvshows` that have at least one genre linked.

- Each record should display: `tv_shows.title - tv_show_genres.genre_id`
- Results must be sorted in ascending order by `tv_shows.title` and `tv_show_genres.genre_id`
- You can use only one `SELECT` statement
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat 8-genre_id_by_show.sql | mysql -hlocalhost -uroot -p hbtn_0d_tvshow
s
Enter password:
title   genre_id
Breaking Bad    1
Breaking Bad    6
Breaking Bad    7
Breaking Bad    8
Dexter  1
Dexter  2
Dexter  6
Dexter  7
Dexter  8
Game of Thrones 1
Game of Thrones 3
Game of Thrones 4
House   1
House   2
New Girl    5
Silicon Valley  5
The Big Bang Theory 5
The Last Man on Earth   1
The Last Man on Earth   5
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `8-genre_id_by_show.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

9. No genre

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Import the database dump from `hbtn_0d_tvshows` to your MySQL server: download (https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_programming+/274/hbtn_0d_tvshows.sql)

Write a script that lists all shows contained in `hbtn_0d_tvshows` without a genre linked.

- Each record should display: `tv_shows.title - tv_show_genres.genre_id`
- Results must be sorted in ascending order by `tv_shows.title` and `tv_show_genres.genre_id`
- You can use only one `SELECT` statement
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat 9-no_genre.sql | mysql -hlocalhost -uroot -p hbtn_0d_tvshows
Enter password:
title   genre_id
Better Call Saul    NULL
Homeland    NULL
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `9-no_genre.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

10. Number of shows by genre

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Import the database dump from `hbtn_0d_tvshows` to your MySQL server: download (https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_programming+/274/hbtn_0d_tvshows.sql)

Write a script that lists all genres from `hbtn_0d_tvshows` and displays the number of shows linked to each.

- Each record should display: `<TV Show genre> - <Number of shows linked to this genre>`
- First column must be called `genre`
- Second column must be called `number_of_shows`
- Don't display a genre that doesn't have any shows linked
- Results must be sorted in descending order by the number of shows linked
- You can use only one `SELECT` statement
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat 10-count_shows_by_genre.sql | mysql -hlocalhost -uroot -p hbtn_0d_t
vshows
Enter password:
genre   number_of_shows
Drama   5
Comedy  4
Mystery 2
Crime   2
Suspense    2
Thriller    2
Adventure   1
Fantasy 1
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `10-count_shows_by_genre.sql`

Review your work | **>_ Get a sandbox** **6/6** pts

11. Rotten tomatoes

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Import the database `hbtn_0d_tvshows_rate` dump to your MySQL server: download (https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_programming+/274/hbtn_0d_tvshows_rate.sql)

Write a script that lists all shows from `hbtn_0d_tvshows_rate` by their rating.

- Each record should display: `tv_shows.title - rating sum`
- Results must be sorted in descending order by the rating
- You can use only one `SELECT` statement
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat 11-rating_shows.sql | mysql -hlocalhost -uroot -p hbtn_0d_tvshows_r
ate
Enter password:
title   rating
Better Call Saul    163
Homeland    145
Silicon Valley  82
Game of Thrones 79
Dexter  24
House   21
Breaking Bad    16
The Last Man on Earth   10
The Big Bang Theory 0
New Girl    0
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `11-rating_shows.sql`

Review your work | >_ Get a sandbox **6/6** pts

(/)

12. Best genre

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Import the database dump from `hbtn_0d_tvshows_rate` to your MySQL server: download (https://s3.amazonaws.com/intranet-projects-files/holbertonschool-higher-level_programming+/274/hbtn_0d_tvshows_rate.sql)

Write a script that lists all genres in the database `hbtn_0d_tvshows_rate` by their rating.

- Each record should display: `tv_genres.name - rating sum`
- Results must be sorted in descending order by their rating
- You can use only one `SELECT` statement
- The database name will be passed as an argument of the `mysql` command

```
guillaume@ubuntu:~/$ cat 12-rating_genres.sql | mysql -hlocalhost -uroot -p hbtn_0d_tvshows_
rate
Enter password:
name    rating
Drama   150
Comedy  92
Adventure   79
Fantasy 79
Mystery 45
Crime   40
Suspense    40
Thriller    40
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `12-rating_genres.sql`

Review your work | >_ Get a sandbox **6/6** pts

13. We are all unique!

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that creates a table `users` following these requirements:

- With these attributes:
  - `id` , integer, never null, auto increment and primary key
  - `email` , string (255 characters), never null and unique
  - `name` , string (255 characters)
- If the table already exists, your script should not fail
- Your script can be executed on any database

Context: *Make an attribute unique directly in the table schema will enforced your business rules and avoid bugs in your application*

```
bob@dylan:~$ echo "SELECT * FROM users;" | mysql -uroot -p holberton
Enter password:
ERROR 1146 (42S02) at line 1: Table 'holberton.users' doesn't exist
bob@dylan:~$
bob@dylan:~$ cat 13-uniq_users.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ echo 'INSERT INTO users (email, name) VALUES ("bob@dylan.com", "Bob");' | mysql
-uroot -p holberton
Enter password:
bob@dylan:~$ echo 'INSERT INTO users (email, name) VALUES ("sylvie@dylan.com", "Sylvie");' |
mysql -uroot -p holberton
Enter password:
bob@dylan:~$ echo 'INSERT INTO users (email, name) VALUES ("bob@dylan.com", "Jean");' | mysq
l -uroot -p holberton
Enter password:
ERROR 1062 (23000) at line 1: Duplicate entry 'bob@dylan.com' for key 'email'
bob@dylan:~$
bob@dylan:~$ echo "SELECT * FROM users;" | mysql -uroot -p holberton
Enter password:
id  email   name
1   bob@dylan.com   Bob
2   sylvie@dylan.com    Sylvie
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `13-uniq_users.sql`

Review your work | **>_ Get a sandbox** **8/8** pts

14. In and not out

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that creates a table `users` following these requirements:

- With these attributes:
  - `id` , integer, never null, auto increment and primary key
  - `email` , string (255 characters), never null and unique
  - `name` , string (255 characters)
  - `country` , enumeration of countries: `US` , `CO` and `TN` , never null (= default will be the first element of the enumeration, here `US`)
- If the table already exists, your script should not fail
- Your script can be executed on any database

```
bob@dylan:~$ echo "SELECT * FROM users;" | mysql -uroot -p holberton
Enter password:
ERROR 1146 (42S02) at line 1: Table 'holberton.users' doesn't exist
bob@dylan:~$
bob@dylan:~$ cat 14-country_users.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ echo 'INSERT INTO users (email, name, country) VALUES ("bob@dylan.com", "Bob",
"US");' | mysql -uroot -p holberton
Enter password:
bob@dylan:~$ echo 'INSERT INTO users (email, name, country) VALUES ("sylvie@dylan.com", "Syl
vie", "CO");' | mysql -uroot -p holberton
Enter password:
bob@dylan:~$ echo 'INSERT INTO users (email, name, country) VALUES ("jean@dylan.com", "Jea
n", "FR");' | mysql -uroot -p holberton
Enter password:
ERROR 1265 (01000) at line 1: Data truncated for column 'country' at row 1
bob@dylan:~$
bob@dylan:~$ echo 'INSERT INTO users (email, name) VALUES ("john@dylan.com", "John");' | mys
ql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ echo "SELECT * FROM users;" | mysql -uroot -p holberton
Enter password:
id  email   name    country
1   bob@dylan.com   Bob US
2   sylvie@dylan.com    Sylvie  CO
3   john@dylan.com  John    US
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `14-country_users.sql`

Review your work | **>_ Get a sandbox** **7/7** pts

15. Best band ever!

mandatory

(/)

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that ranks country origins of bands, ordered by the number of (non-unique) fans

Requirements:

- Import this table dump: metal_bands.sql.zip (https://s3.amazonaws.com/alu-intranet.hbtn.io/uploads/misc/2020/6/ab2979f058de215f0f2ae5b052739e76d3c02ac5.zip?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIARDDGGGOUZTW2RLVB%2F20241102%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20241102T173707Z&X-Amz-Expires=345600&X-Amz-SignedHeaders=host&X-Amz-Signature=0983956a822a786caae2231753c6d19e911b4273092926d1336cc83f00546aa9)
- Column names must be: `origin` and `nb_fans`
- Your script can be executed on any database

Context: *Calculate/compute something is always power intensive... better to distribute the load!*

```
bob@dylan:~$ cat metal_bands.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 15-fans.sql | mysql -uroot -p holberton > tmp_res ; head tmp_res
Enter password:
origin  nb_fans
USA 99349
Sweden  47169
Finland 32878
United Kingdom  32518
Germany 29486
Norway  22405
Canada  8874
The Netherlands 8819
Italy   7178
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `15-fans.sql`

Review your work | >_ Get a sandbox

4/4 pts

16. Old school band

mandatory

Write a SQL script that lists all bands with `Glam rock` as their main style, ranked by their longevity

Requirements:

- Import this table dump: metal_bands.sql.zip (https://s3.amazonaws.com/alu-intranet.hbtn.io/uploads/misc/2020/6/ab2979f058de215f0f2ae5b052739e76d3c02ac5.zip?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIARDDGGGOUZTW2RLVB%2F20241102%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20241102T173707Z&X-Amz-Expires=345600&X-Amz-SignedHeaders=host&X-Amz-Signature=0983956a822a786caae2231753c6d19e911b4273092926d1336cc83f00546aa9)
- Column names must be:
  - `band_name`
  - `lifespan` until 2020 (in years)
- You should use attributes `formed` and `split` for computing the `lifespan`
- Your script can be executed on any database

```
bob@dylan:~$ cat metal_bands.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 16-glam_rock.sql | mysql -uroot -p holberton
Enter password:
band_name   lifespan
Alice Cooper    56
Mötley Crüe   34
Marilyn Manson  31
The 69 Eyes 30
Hardcore Superstar  23
Nasty Idols 0
Hanoi Rocks 0
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `16-glam_rock.sql`

Review your work | **>_ Get a sandbox** **4/4** pts

17. Buy buy buy

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that creates a trigger that decreases the quantity of an item after adding a new order.

Quantity in the table `items` can be negative.

Context: *Updating multiple tables for one action from your application can generate issue: network disconnection, crash, etc... to keep your data in a good shape, let MySQL do it for you!*

(*/*)

```
bob@dylan:~$ cat 17-init.sql
-- Initial
DROP TABLE IF EXISTS items;
DROP TABLE IF EXISTS orders;

CREATE TABLE IF NOT EXISTS items (
    name VARCHAR(255) NOT NULL,
    quantity int NOT NULL DEFAULT 10
);

CREATE TABLE IF NOT EXISTS orders (
    item_name VARCHAR(255) NOT NULL,
    number int NOT NULL
);

INSERT INTO items (name) VALUES ("apple"), ("pineapple"), ("pear");

bob@dylan:~$
bob@dylan:~$ cat 17-init.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 17-store.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 17-main.sql
Enter password:
-- Show and add orders
SELECT * FROM items;
SELECT * FROM orders;

INSERT INTO orders (item_name, number) VALUES ('apple', 1);
INSERT INTO orders (item_name, number) VALUES ('apple', 3);
INSERT INTO orders (item_name, number) VALUES ('pear', 2);

SELECT "--";

SELECT * FROM items;
SELECT * FROM orders;

bob@dylan:~$
bob@dylan:~$ cat 17-main.sql | mysql -uroot -p holberton
Enter password:
name    quantity
apple   10
pineapple   10
pear    10
--
--
name    quantity
apple   6
pineapple   10
pear    8
```

```
item_name   number
apple   1
apple   3
pear    2
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `17-store.sql`

Review your work | **>_ Get a sandbox** **4/4** pts

18. Email validation to sent

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that creates a trigger that resets the attribute `valid_email` only when the `email` has been changed.

Context: *Nothing related to MySQL, but perfect for user email validation - distribute the logic to the database itself!*

```
bob@dylan:~$ cat 18-init.sql
-- Initial
DROP TABLE IF EXISTS users;

CREATE TABLE IF NOT EXISTS users (
    id int not null AUTO_INCREMENT,
    email varchar(255) not null,
    name varchar(255),
    valid_email boolean not null default 0,
    PRIMARY KEY (id)
);

INSERT INTO users (email, name) VALUES ("bob@dylan.com", "Bob");
INSERT INTO users (email, name, valid_email) VALUES ("sylvie@dylan.com", "Sylvie", 1);
INSERT INTO users (email, name, valid_email) VALUES ("jeanne@dylan.com", "Jeanne", 1);

bob@dylan:~$
bob@dylan:~$ cat 18-init.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 18-valid_email.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 18-main.sql
Enter password:
-- Show users and update (or not) email
SELECT * FROM users;

UPDATE users SET valid_email = 1 WHERE email = "bob@dylan.com";
UPDATE users SET email = "sylvie+new@dylan.com" WHERE email = "sylvie@dylan.com";
UPDATE users SET name = "Jannis" WHERE email = "jeanne@dylan.com";

SELECT "--";
SELECT * FROM users;

UPDATE users SET email = "bob@dylan.com" WHERE email = "bob@dylan.com";

SELECT "--";
SELECT * FROM users;

bob@dylan:~$
bob@dylan:~$ cat 18-main.sql | mysql -uroot -p holberton
Enter password:
id  email   name    valid_email
1   bob@dylan.com   Bob 0
2   sylvie@dylan.com    Sylvie  1
3   jeanne@dylan.com    Jeanne  1
--
--
id  email   name    valid_email
1   bob@dylan.com   Bob 1
2   sylvie+new@dylan.com    Sylvie  0
```

```
3   jeanne@dylan.com    Jannis  1
--
--
id email   name    valid_email
1   bob@dylan.com   Bob 1
2   sylvie+new@dylan.com    Sylvie  0
3   jeanne@dylan.com    Jannis  1
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `18-valid_email.sql`

Review your work | **>_ Get a sandbox** **4/4** pts

19. Add bonus

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that creates a stored procedure `AddBonus` that adds a new correction for a student.

Requirements:

- Procedure `AddBonus` is taking 3 inputs (in this order):
  - `user_id`, a `users.id` value (you can assume `user_id` is linked to an existing `users`)
  - `project_name`, a new or already exists `projects` - if no `projects.name` found in the table, you should create it
  - `score`, the score value for the correction

Context: *Write code in SQL is a nice level up!*

```
bob@dylan:~$ cat 19-init.sql
-- Initial
DROP TABLE IF EXISTS corrections;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS projects;

CREATE TABLE IF NOT EXISTS users (
    id int not null AUTO_INCREMENT,
    name varchar(255) not null,
    average_score float default 0,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS projects (
    id int not null AUTO_INCREMENT,
    name varchar(255) not null,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS corrections (
    user_id int not null,
    project_id int not null,
    score int default 0,
    KEY `user_id` (`user_id`),
    KEY `project_id` (`project_id`),
    CONSTRAINT fk_user_id FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCAD
E,
    CONSTRAINT fk_project_id FOREIGN KEY (`project_id`) REFERENCES `projects` (`id`) ON DELE
TE CASCADE
);

INSERT INTO users (name) VALUES ("Bob");
SET @user_bob = LAST_INSERT_ID();

INSERT INTO users (name) VALUES ("Jeanne");
SET @user_jeanne = LAST_INSERT_ID();

INSERT INTO projects (name) VALUES ("C is fun");
SET @project_c = LAST_INSERT_ID();

INSERT INTO projects (name) VALUES ("Python is cool");
SET @project_py = LAST_INSERT_ID();


INSERT INTO corrections (user_id, project_id, score) VALUES (@user_bob, @project_c, 80);
INSERT INTO corrections (user_id, project_id, score) VALUES (@user_bob, @project_py, 96);

INSERT INTO corrections (user_id, project_id, score) VALUES (@user_jeanne, @project_c, 91);
INSERT INTO corrections (user_id, project_id, score) VALUES (@user_jeanne, @project_py, 73);

bob@dylan:~$
bob@dylan:~$ cat 19-init.sql | mysql -uroot -p holberton
```

```
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 19-bonus.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 19-main.sql
Enter password:
-- Show and add bonus correction
SELECT * FROM projects;
SELECT * FROM corrections;

SELECT "--";

CALL AddBonus((SELECT id FROM users WHERE name = "Jeanne"), "Python is cool", 100);

CALL AddBonus((SELECT id FROM users WHERE name = "Jeanne"), "Bonus project", 100);
CALL AddBonus((SELECT id FROM users WHERE name = "Bob"), "Bonus project", 10);

CALL AddBonus((SELECT id FROM users WHERE name = "Jeanne"), "New bonus", 90);

SELECT "--";

SELECT * FROM projects;
SELECT * FROM corrections;

bob@dylan:~$
bob@dylan:~$ cat 19-main.sql | mysql -uroot -p holberton
Enter password:
id  name
1   C is fun
2   Python is cool
user_id project_id  score
1   1   80
1   2   96
2   1   91
2   2   73
--
--
--
--
id  name
1   C is fun
2   Python is cool
3   Bonus project
4   New bonus
user_id project_id  score
1   1   80
1   2   96
2   1   91
2   2   73
2   2   100
2   3   100
```

```
1   3   10
2   4   90
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `19-bonus.sql`

Review your work | **>_ Get a sandbox** **4/4** pts

20. Average score

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that creates a stored procedure `ComputeAverageScoreForUser` that computes and store the average score for a student.

Requirements:

- Procedure `ComputeAverageScoreForUser` is taking 1 input:
  - `user_id`, a `users.id` value (you can assume `user_id` is linked to an existing `users`)

```
bob@dylan:~$ cat 20-init.sql
-- Initial
DROP TABLE IF EXISTS corrections;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS projects;

CREATE TABLE IF NOT EXISTS users (
    id int not null AUTO_INCREMENT,
    name varchar(255) not null,
    average_score float default 0,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS projects (
    id int not null AUTO_INCREMENT,
    name varchar(255) not null,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS corrections (
    user_id int not null,
    project_id int not null,
    score int default 0,
    KEY `user_id` (`user_id`),
    KEY `project_id` (`project_id`),
    CONSTRAINT fk_user_id FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCAD
E,
    CONSTRAINT fk_project_id FOREIGN KEY (`project_id`) REFERENCES `projects` (`id`) ON DELE
TE CASCADE
);

INSERT INTO users (name) VALUES ("Bob");
SET @user_bob = LAST_INSERT_ID();

INSERT INTO users (name) VALUES ("Jeanne");
SET @user_jeanne = LAST_INSERT_ID();

INSERT INTO projects (name) VALUES ("C is fun");
SET @project_c = LAST_INSERT_ID();

INSERT INTO projects (name) VALUES ("Python is cool");
SET @project_py = LAST_INSERT_ID();


INSERT INTO corrections (user_id, project_id, score) VALUES (@user_bob, @project_c, 80);
INSERT INTO corrections (user_id, project_id, score) VALUES (@user_bob, @project_py, 96);

INSERT INTO corrections (user_id, project_id, score) VALUES (@user_jeanne, @project_c, 91);
INSERT INTO corrections (user_id, project_id, score) VALUES (@user_jeanne, @project_py, 73);

bob@dylan:~$
bob@dylan:~$ cat 20-init.sql | mysql -uroot -p holberton
```

```
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 20-average_score.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 20-main.sql
-- Show and compute average score
SELECT * FROM users;
SELECT * FROM corrections;

SELECT "--";
CALL ComputeAverageScoreForUser((SELECT id FROM users WHERE name = "Jeanne"));

SELECT "--";
SELECT * FROM users;

bob@dylan:~$
bob@dylan:~$ cat 20-main.sql | mysql -uroot -p holberton
Enter password:
id  name    average_score
1   Bob 0
2   Jeanne  0
user_id project_id  score
1   1   80
1   2   96
2   1   91
2   2   73
--
--
--
--
id  name    average_score
1   Bob 0
2   Jeanne  82
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `20-average_score.sql`

Review your work | **>_ Get a sandbox** **4/4** pts

21. Safe divide

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a SQL script that creates a function `SafeDiv` that divides (and returns) the first by the second number or returns 0 if the second number is equal to 0.

Requirements:

- You must create a function
- The function `SafeDiv` takes 2 arguments:
  - `a`, INT
  - `b`, INT
- And returns `a / b` or 0 if `b == 0`

```
bob@dylan:~$ cat 21-init.sql
-- Initial
DROP TABLE IF EXISTS numbers;

CREATE TABLE IF NOT EXISTS numbers (
    a int default 0,
    b int default 0
);

INSERT INTO numbers (a, b) VALUES (10, 2);
INSERT INTO numbers (a, b) VALUES (4, 5);
INSERT INTO numbers (a, b) VALUES (2, 3);
INSERT INTO numbers (a, b) VALUES (6, 3);
INSERT INTO numbers (a, b) VALUES (7, 0);
INSERT INTO numbers (a, b) VALUES (6, 8);

bob@dylan:~$ cat 21-init.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ cat 21-div.sql | mysql -uroot -p holberton
Enter password:
bob@dylan:~$
bob@dylan:~$ echo "SELECT (a / b) FROM numbers;" | mysql -uroot -p holberton
Enter password:
(a / b)
5.0000
0.8000
0.6667
2.0000
NULL
0.7500
bob@dylan:~$
bob@dylan:~$ echo "SELECT SafeDiv(a, b) FROM numbers;" | mysql -uroot -p holberton
Enter password:
SafeDiv(a, b)
5
0.800000011920929
0.6666666865348816
2
0
0.75
bob@dylan:~$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `21-div.sql`

Review your work | >_ Get a sandbox **4/4** pts

(/)

22. List all databases

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that lists all databases in MongoDB.

```
guillaume@ubuntu:~/$ cat 22-list_databases | mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
admin        0.000GB
config       0.000GB
local        0.000GB
logs         0.005GB
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `22-list_databases`

Review your work | >_ Get a sandbox **9/9** pts

23. Create a database

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that creates or uses the database `my_db` :

```
guillaume@ubuntu:~/$ cat 22-list_databases | mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
admin        0.000GB
config       0.000GB
local        0.000GB
logs         0.005GB
bye
guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ cat 23-use_or_create_database | mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
switched to db my_db
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `23-use_or_create_database`

Review your work | **>_ Get a sandbox** **8/8** pts

24. Insert document

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that inserts a document in the collection `school`:

- The document must have one attribute `name` with value “Holberton school”
- The database name will be passed as option of `mongo` command

```
guillaume@ubuntu:~/$ cat 24-insert | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
WriteResult({ "nInserted" : 1 })
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`

- Directory: `pipeline/databases`
- File: `24-insert`

(/)

Review your work | **>_ Get a sandbox** **8/8** pts

25. All documents

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that lists all documents in the collection `school`:

- The database name will be passed as option of `mongo` command

```
guillaume@ubuntu:~/$ cat 25-all | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
{ "_id" : ObjectId("5a8fad532b69437b63252406"), "name" : "Holberton school" }
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `25-all`

Review your work | **>_ Get a sandbox** **9/9** pts

26. All matches

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that lists all documents with `name="Holberton school"` in the collection `school`:

- The database name will be passed as option of `mongo` command

```
guillaume@ubuntu:~/$ cat 26-match | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
{ "_id" : ObjectId("5a8fad532b69437b63252406"), "name" : "Holberton school" }
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `26-match`

Review your work | **>_ Get a sandbox** **11/11** pts

27. Count

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that displays the number of documents in the collection `school`:

- The database name will be passed as option of `mongo` command

```
guillaume@ubuntu:~/$ cat 27-count | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
1
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `27-count`

Review your work | **>_ Get a sandbox** **9/9** pts

28. Update

mandatory

(/)

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that adds a new attribute to a document in the collection `school`:

- The script should update only document with `name="Holberton school"` (all of them)
- The update should add the attribute `address` with the value “972 Mission street”
- The database name will be passed as option of `mongo` command

```
guillaume@ubuntu:~/$ cat 28-update | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
bye
guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ cat 26-match | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
{ "_id" : ObjectId("5a8fad532b69437b63252406"), "name" : "Holberton school", "address" : "97
2 Mission street" }
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `28-update`

Review your work | **>_ Get a sandbox** **11/11** pts

29. Delete by match

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that deletes all documents with `name="Holberton school"` in the collection `school`:

- The database name will be passed as option of `mongo` command

```
guillaume@ubuntu:~/$ cat 29-delete | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
{ "acknowledged" : true, "deletedCount" : 1 }
bye
guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ cat 26-match | mongo my_db
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/my_db
MongoDB server version: 3.6.3
bye
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `29-delete`

Review your work | **>_ Get a sandbox** **11/11** pts

30. List all documents in Python

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a Python function that lists all documents in a collection:

- Prototype: `def list_all(mongo_collection):`
- Return an empty list if no document in the collection
- `mongo_collection` will be the `pymongo` collection object

```
guillaume@ubuntu:~/$ cat 30-main.py
#!/usr/bin/env python3
""" 30-main """
from pymongo import MongoClient
list_all = __import__('30-all').list_all

if __name__ == "__main__":
    client = MongoClient('mongodb://127.0.0.1:27017')
    school_collection = client.my_db.school
    schools = list_all(school_collection)
    for school in schools:
        print("[{}] {}".format(school.get('_id'), school.get('name')))

guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ ./30-main.py
[5a8f60cfd4321e1403ba7ab9] Holberton school
[5a8f60cfd4321e1403ba7aba] UCSD
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `30-all.py`

Review your work | **>_ Get a sandbox** **9/9** pts

31. Insert a document in Python

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a Python function that inserts a new document in a collection based on `kwargs`:

- Prototype: `def insert_school(mongo_collection, **kwargs):`
- `mongo_collection` will be the `pymongo` collection object
- Returns the new `_id`

```
guillaume@ubuntu:~/$ cat 31-main.py
#!/usr/bin/env python3
""" 31-main """
from pymongo import MongoClient
list_all = __import__('30-all').list_all
insert_school = __import__('31-insert_school').insert_school

if __name__ == "__main__":
    client = MongoClient('mongodb://127.0.0.1:27017')
    school_collection = client.my_db.school
    new_school_id = insert_school(school_collection, name="UCSF", address="505 Parnassus Av
e")
    print("New school created: {}".format(new_school_id))

    schools = list_all(school_collection)
    for school in schools:
        print("[{}] {} {}".format(school.get('_id'), school.get('name'), school.get('addres
s', "")))

guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ ./31-main.py
New school created: 5a8f60cfd4321e1403ba7abb
[5a8f60cfd4321e1403ba7ab9] Holberton school
[5a8f60cfd4321e1403ba7aba] UCSD
[5a8f60cfd4321e1403ba7abb] UCSF 505 Parnassus Ave
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `31-insert_school.py`

Review your work | **>_ Get a sandbox** **14/14** pts

32. Change school topics

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a Python function that changes all topics of a school document based on the name:

- Prototype: `def update_topics(mongo_collection, name, topics):`
- `mongo_collection` will be the `pymongo` collection object
- `name` (string) will be the school name to update
- `topics` (list of strings) will be the list of topics approached in the school

```
guillaume@ubuntu:~/$ cat 32-main.py
#!/usr/bin/env python3
""" 32-main """
from pymongo import MongoClient
list_all = __import__('30-all').list_all
update_topics = __import__('32-update_topics').update_topics

if __name__ == "__main__":
    client = MongoClient('mongodb://127.0.0.1:27017')
    school_collection = client.my_db.school
    update_topics(school_collection, "Holberton school", ["Sys admin", "AI", "Algorithm"])

    schools = list_all(school_collection)
    for school in schools:
        print("[{}] {} {}".format(school.get('_id'), school.get('name'), school.get('topic
s', "")))

    update_topics(school_collection, "Holberton school", ["iOS"])

    schools = list_all(school_collection)
    for school in schools:
        print("[{}] {} {}".format(school.get('_id'), school.get('name'), school.get('topic
s', "")))

guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ ./32-main.py
[5a8f60cfd4321e1403ba7abb] UCSF
[5a8f60cfd4321e1403ba7aba] UCSD
[5a8f60cfd4321e1403ba7ab9] Holberton school ['Sys admin', 'AI', 'Algorithm']
[5a8f60cfd4321e1403ba7abb] UCSF
[5a8f60cfd4321e1403ba7aba] UCSD
[5a8f60cfd4321e1403ba7ab9] Holberton school ['iOS']
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `32-update_topics.py`

Review your work | **>_ Get a sandbox** **11/11** pts

33. Where can I learn Python?

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a Python function that returns the list of school having a specific topic:

- Prototype: `def schools_by_topic(mongo_collection, topic):`
- `mongo_collection` will be the `pymongo` collection object
- `topic` (string) will be topic searched

```
guillaume@ubuntu:~/$ cat 33-main.py
#!/usr/bin/env python3
""" 33-main """
from pymongo import MongoClient
list_all = __import__('30-all').list_all
insert_school = __import__('31-insert_school').insert_school
schools_by_topic = __import__('33-schools_by_topic').schools_by_topic

if __name__ == "__main__":
    client = MongoClient('mongodb://127.0.0.1:27017')
    school_collection = client.my_db.school

    j_schools = [
        { 'name': "Holberton school", 'topics': ["Algo", "C", "Python", "React"]},
        { 'name': "UCSF", 'topics': ["Algo", "MongoDB"]},
        { 'name': "UCLA", 'topics': ["C", "Python"]},
        { 'name': "UCSD", 'topics': ["Cassandra"]},
        { 'name': "Stanford", 'topics': ["C", "React", "Javascript"]}
    ]
    for j_school in j_schools:
        insert_school(school_collection, **j_school)

    schools = schools_by_topic(school_collection, "Python")
    for school in schools:
        print("[{}] {} {}".format(school.get('_id'), school.get('name'), school.get('topic
s', "")))

guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ ./33-main.py
[5a90731fd4321e1e5a3f53e3] Holberton school ['Algo', 'C', 'Python', 'React']
[5a90731fd4321e1e5a3f53e5] UCLA ['C', 'Python']
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `33-schools_by_topic.py`

Review your work | **>_ Get a sandbox** **13/13** pts

34. Log stats

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a Python script that provides some stats about Nginx logs stored in MongoDB:

- Database: `logs`
- Collection: `nginx`
- Display (same as the example):
  - first line: `x logs` where `x` is the number of documents in this collection
  - second line: `Methods:`
  - 5 lines with the number of documents with the `method = ["GET", "POST", "PUT", "PATCH", "DELETE"]` in this order (see example below - warning: it's a tabulation before each line)
  - one line with the number of documents with:
    - `method=GET`
    - `path=/status`

You can use this dump as data sample: dump.zip (https://s3.amazonaws.com/alu-intranet.hbtn.io/uploads/misc/2020/6/645541f867bb79ae47b7a80922e9a48604a569b9.zip?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIARDDGGGOUZTW2RLVB%2F20241102%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20241102T173707Z&X-Amz-Expires=345600&X-Amz-SignedHeaders=host&X-Amz-Signature=5364d4fde7f97a6c625b8379ff56e3a544680770ab1d9c80b865eef05967915c)

The output of your script **must be exactly the same as the example**

```
guillaume@ubuntu:~/$ curl -o dump.zip -s "https://s3.amazonaws.com/intranet-projects-files/holbertonschool-webstack/411/dump.zip"
guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ unzip dump.zip
Archive:  dump.zip
   creating: dump/
   creating: dump/logs/
  inflating: dump/logs/nginx.metadata.json
  inflating: dump/logs/nginx.bson
guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ mongorestore dump
2018-02-23T20:12:37.807+0000    preparing collections to restore from
2018-02-23T20:12:37.816+0000    reading metadata for logs.nginx from dump/logs/nginx.metadata.json
2018-02-23T20:12:37.825+0000    restoring logs.nginx from dump/logs/nginx.bson
2018-02-23T20:12:40.804+0000    [##......................]  logs.nginx  1.21MB/13.4MB  (9.0%)
2018-02-23T20:12:43.803+0000    [#####...................]  logs.nginx  2.88MB/13.4MB  (21.4%)
2018-02-23T20:12:46.803+0000    [#######.................]  logs.nginx  4.22MB/13.4MB  (31.4%)
2018-02-23T20:12:49.803+0000    [##########..............]  logs.nginx  5.73MB/13.4MB  (42.7%)
2018-02-23T20:12:52.803+0000    [############............]  logs.nginx  7.23MB/13.4MB  (53.8%)
2018-02-23T20:12:55.803+0000    [###############.........]  logs.nginx  8.53MB/13.4MB  (63.5%)
2018-02-23T20:12:58.803+0000    [#################.......]  logs.nginx  10.1MB/13.4MB  (74.9%)
2018-02-23T20:13:01.803+0000    [####################....]  logs.nginx  11.3MB/13.4MB  (83.9%)
2018-02-23T20:13:04.803+0000    [######################..]  logs.nginx  12.8MB/13.4MB  (94.9%)
2018-02-23T20:13:06.228+0000    [########################]  logs.nginx  13.4MB/13.4MB  (100.0%)
2018-02-23T20:13:06.230+0000    no indexes to restore
2018-02-23T20:13:06.231+0000    finished restoring logs.nginx (94778 documents)
2018-02-23T20:13:06.232+0000    done
guillaume@ubuntu:~/$
guillaume@ubuntu:~/$ ./34-log_stats.py
94778 logs
Methods:
    method GET: 93842
    method POST: 229
    method PUT: 0
    method PATCH: 0
    method DELETE: 0
47415 status check
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alu-machine_learning`
- Directory: `pipeline/databases`
- File: `34-log_stats.py`

Review your work | >_ Get a sandbox

12/12 pts

Done with the mandatory tasks? Unlock 7 advanced tasks now!

Score

100%

Congratulations! You made it!

The next project will be available on Sunday, Nov 3rd.

Go to home (/)

Previous project (/projects/2372)