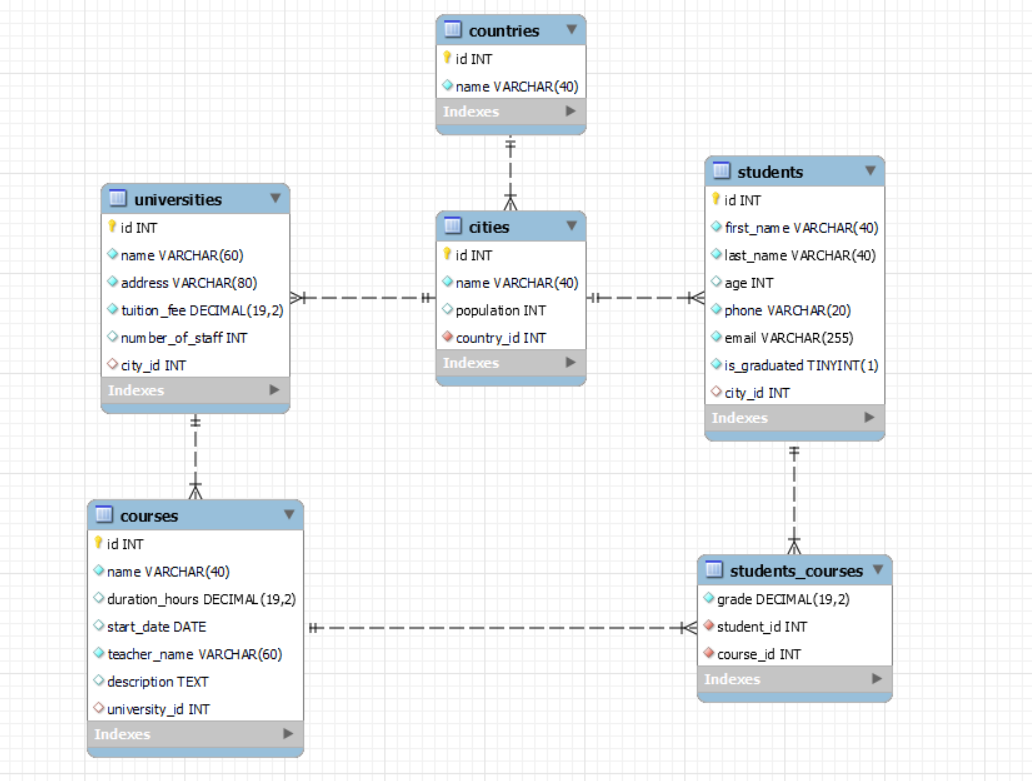
# MySQL Exam Universities DB

*The Universities DB is designed to make a connection between universities for exchange programs, information and help students, parents, and academic institutions make informed decisions about their education choices. The system provides users with easy access to information on universities and their courses, enabling them to compare different options and make informed decisions.*

Help by implementing the database structure, optimize the system and make this powerful tool for students, and researchers worldwide, enabling them to make informed decisions about higher education, career choices, and research opportunities.

## Section 0: Database Overview

You have been given an Entity / Relationship Diagram of the Database:



The **universities\_db** needs to hold information about **countries**, **cities**, **universities**, **students**, **courses**.

Your task is to create a database called **universities\_db**. Then you will have to create several **tables**.

* countries – contains information about the **countries**.
* cities – contains information about the **cities**.
  + Each **city** has a **country.**
* universities – contains information about the **universities**.
  + Each university has a city.
* students – contains information about the **students**.
  + Each student has a **city**.
* courses – contains information about the **courses**.
  + Each course has a **university**.
* students\_courses – a **many** to **many** **mapping** table between the **students** and the **courses**.

## Section 1: Data Definition Language (DDL) – 40 pts

Make sure you implement the whole database correctly on your local machine, so that you could work with it.

The instructions you'll be given will be the minimal needed for you to implement the database.

### Table Design

You have been tasked to create the tables in the database by the following models:

**countries**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |

**cities**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| population | **Integer,** from **1** to **2,147,483,647.** |  |
| country\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table countries.  **NULL** is **NOT** permitted**.** |

**universities**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **60 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| address | A **string** containing a maximum of **80 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| tuition\_fee | **DECIMAL**, up to **19 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |
| number\_of\_staff | **Integer,** from **1** to **2,147,483,647.** |  |
| city­\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table cities. |

**students**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| first\_name | A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| last\_name | A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| age | **Integer,** from **1** to **2,147,483,647.** |  |
| phone | A **string** containing a maximum of **20 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| email | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| is\_graduated | Can be **true** or **false**. | **NULL** is **NOT** permitted**.** |
| city\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table cities. |

**courses**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **40 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| duration\_hours | **DECIMAL**, up to **19 digits**, **2** of which after the **decimal point**. |  |
| start\_date | The starting **date** of the course. |  |
| teacher\_name | A **string** containing a maximum of **60 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| description | A very **long** string field |  |
| university\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table universities. |

**students\_courses**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| grade | **DECIMAL**, up to **19 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |
| student\_id | **Integer**, from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted.  Relationship with table students. |
| course\_id | **Integer**, from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted.  Relationship with table courses. |

Submit your solutions in Judge on the first task. Submit **all** SQL table creation statements.

You will also be given a data.sql file. It will contain a **dataset** with random data which you will need to **store** in your **local database**. This data will be given to you so you will not have to think of data and lose essential time in the process. The data is in the form of **INSERT** statement queries.

## Section 2: Data Manipulation Language (DML) – 30 pts

Here we need to do several manipulations in the database, like changing data, adding data etc.

### Insert

You will have to **insert** records of data into the **courses** table, based on the **courses** table.

For **courses** with **id** equal or lesser than **5**, **insert data** in the **course** table with the **following values**:

• **name** – set it to the **teacher name** followed by white space and then "course"

*(teacher\_name + " " + "course")*

• **duration\_hours** – set it to the total number of characters from the **course name** and the result **devided** by **10**.

• **start\_date** – set it to the **start date** of the course but **5** days later.

• **teacher\_name** – set it to the **teacher name** but **reversed**.

• **description** – set it to "Course " followed by the **teacher name** and the **description** but **reversed**.

*("Course " + teacher\_name + description\_reversed)*

• **university­\_id** – set it to the **day** of the **start date** of the original **course**.

### Update

Due to inflation and the rising cost of living some universities must raise their tuition fees.

Raise the **tuition fee** by 300 for all **universities** with **id** equal or greater than 5 and less than 12 (**inclusive**).

### Delete

There are some minor bugs in the system and some universities didn’t send the correct information. **Delete** all **universities** for which we don’t have information about the **number of staff.**

## Section 3: Querying – 50 pts

And now we need to do some data extraction. **Note** that the **example results** from **this section** use a **fresh database**. It is **highly recommended** that you **clear** the **database** that has been **manipulated** by the **previous problems** from the **DML** **section** and **insert again** the **dataset** you’ve been given, to ensure **maximum consistency** with the **examples** given in this section.

### Cities

Extract from the **universities\_db** system database, info about the **cities**.

**Order** the results by **population** in **descending** order;

#### Required Columns

* id
* name
* population
* country\_id

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **population** | **country\_id** |
| 14 | Shanghai | 24256800 | 7 |
| 15 | Beijing | 21516000 | 7 |
| ... | ... | ... | ... |
| 7 | Marseille | 852516 | 4 |

### Students age

Write a query that returns: **first\_name**, **last\_name**, **age**, **phone** and **email** from table **students**. **Filter** students with **age** equal or higher than 21.

**Order** the results **descending** by **first\_name**, then by **email ascending**, then by **id** in **ascending** order and show only the first **10** results.

#### Required Columns

* first\_name
* last\_name
* age
* phone
* email

#### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **first\_name** | **last\_name** | **age** | **phone** | **email** |
| William | Mitchell | 21 | 555-945 | william.mitchell@example.com |
| Samantha | Smith | 23 | 555-0034 | sam.smith@example.com |
| Samantha | Lee | 22 | +1-555-5678 | samantha.lee@email.com |
| . . . | . . . | … | . . . | . . . |
| Mia | Moore | 24 | 555-5658 | mia.moore@example.com |

### New students

There are some students that are not signed up for any course but want to be registered in the system with accounts. To find the account details write a query that returns: **full\_name, username** and **password** for all **students** who do not have any assigned **course**. The **full\_name** is their **first\_name** and **last\_name** separated by whitespace. The **username** is generated by using **10** characters from their **email** starting from the **2nd** letter. The **password** is their **phone** **number** but **reversed**. 

**Order** by **password** in **descending** order.

#### Required Columns

* full\_name (first\_name + " " + last\_name)
* username (10 characters long starting from the 2nd)
* password (phone number but reversed)

#### Example

|  |  |  |
| --- | --- | --- |
| **full\_name** | **username** | **password** |
| Avery Martinez | very.marti | 8265-555-1+ |
| Michael Jones | ichael.jon | 7654-555 |
| Oliver Nguyen | liver.nguy | 2209-555-1+ |
| William Tan | illiam.tan | 092765432+ |

### Students count

Every university has courses with students. The directors of every university want to know the total number of students assigned to courses. Extract from the database the **students\_count** (total number of assigned students) in each **university** and the corresponding **university\_name**. Get only those universities with **students\_count** equal or greater than 8.

**Order** the results **descending** by **students\_count** and then by **university\_name** in **descending** order**.**

#### Required Columns

* students\_count
* university\_name

#### Examples

|  |  |
| --- | --- |
| **students\_count** | **university\_name** |
| 15 | Haus und Landwirtschaftliche Schule |
| 12 | Penn |
| 11 | Nikolaus-von-Kues-Gymnasium |
| 10 | Lycée Thiers |
| . . . | . . . |
| 8 | McGill University |

### Price rankings

Make it easier for students when they are searching for a new university. From the database extract the **university\_name**, **city\_name**, **address**, **price\_rank** and **tuition\_fee**. If the **tuition fee** is less then **800** (exclusive) the user must see "**cheap**", equal or above **800** and less than **1200** it should display "**normal**", equal or above **1200** and less than **2500** it should display "**high**" and equal or above that it should display "**expensive**". 

**Order** the results **ascending** by **tuition\_fee**.

#### Required Columns

* university\_name
* city\_name
* address
* price\_rank *(less than 800 – "cheap", equal or above 800 and less than 1200 – "normal", equal or above 1200 and less than 2500 – "high", equal or above 2500 – "expensive)*
* tuition\_fee

#### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **university\_name** | **city\_name** | **address** | **price\_rank** | **tuition\_fee** |
| Universidad Santa Fe | Mexico City | ANGELA BARRIENTOS NO. 101 1ER PISO | cheap | 287.70 |
| Lycée Alain | Marseille | 22 Square de la Couronne | cheap | 421.60 |
| . . . | . . . | . . . | . . . | . . . |
| Penn | New York | 6 West Plaza | expensive | 3883.50 |
| Tongji University | Hangzhou | i Lin Sheng Yan Ji Shi Ju Zi Jie Da Yu Hua Yuan 4dong 401 | expensive | 3904.70 |

## Section 4: Programmability – 30 pts

The time has come for you to prove that you can be a little more dynamic on the database. So, you will have to write several procedures.

### Average grades

Create a **user defined function** with the name **udf\_average\_alumni\_grade\_by\_course\_name(course\_name VARCHAR(60))** that receives a **course name** and returns the **average grades** of the grades from those students that are **graduated**.

#### Required Columns

* course\_name
* average\_alumni\_grade (udf\_average\_alumni\_grade\_by\_course\_name)

#### Example

|  |  |  |
| --- | --- | --- |
| **Query** | | |
| SELECT c.name, udf\_average\_alumni\_grade\_by\_course\_name('Quantum Physics') as average\_alumni\_grade FROM courses c  WHERE c.name = 'Quantum Physics'; | | |
| course\_name | average\_alumni\_grade |  |
| Quantum Physics | 5.80 |  |

### Graduate students

Create a stored procedure udp\_graduate\_all\_students\_by\_year which accepts the following parameters:

* year\_started INT

Extracts data about all **courses** that started on the given **year,** find theassigned **students** and change their **graduated** status to true.

#### Result

|  |
| --- |
| **Query** |
| CALL udp\_graduate\_all\_students\_by\_year(2017); |
| This execution will update 8 students – Emily Wong, Luke Singh, Samtha Fernandez,… |
| Result |
| Emili Wong – is\_graduated(0) -> is\_graduated(1)  Luke Singh – is\_graduated(0) -> is\_graduated(1)  Samantha Fernandeez – is\_graduated(0) -> is\_graduated(1)  … … … |