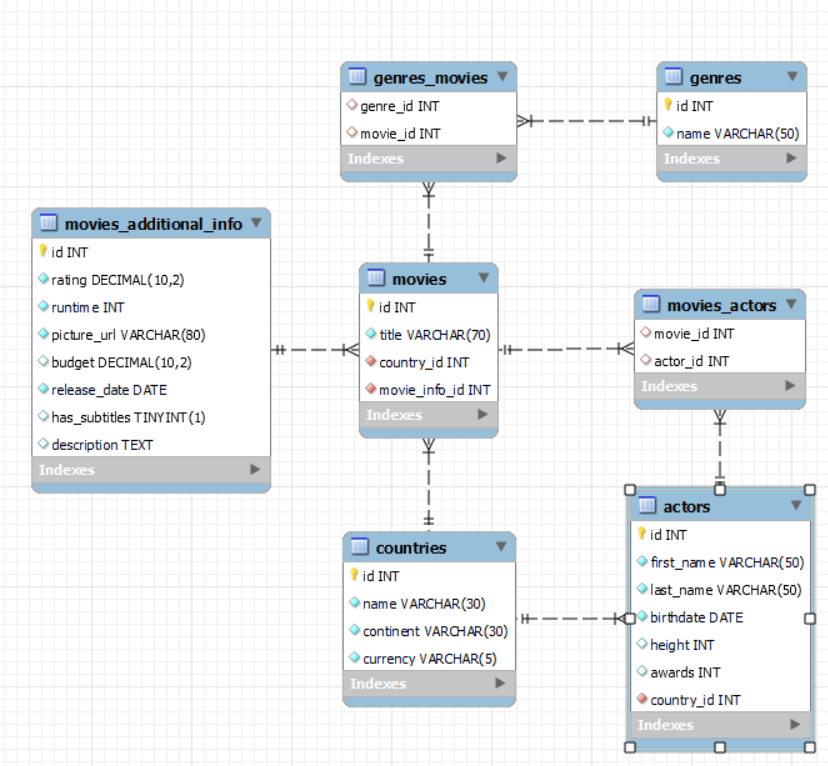
# MySQL Exam SoftUni Internet Movie Database – SU­-IMDb

Exam for the ["MySQL" course @ SoftUni](https://judge.softuni.org/Contests/3315/Database-Basics-MySQL-Retake-Exam-10-April-2022).

The biggest international movie festival is about to begin. They hired a team of programmers to help manage their database. Now you are the leader of the team and you need to manage the system so they can keep track of all movies and actors and finally on the ceremony to give the annual awards to the winners.

## Section 0: Database Overview

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



The **softuni\_imdb’s** **Database** needs to hold information about **movies**, **countries**, **actors**, **genres** and **movies additional info**.

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

* **countries** – contains information about the **countries**.
* **movies** – contains information about the **movies**.
  + Each movie has **actors, country** and **genres**.
* **actors** – contains information about the **actors**.
  + Each actor has a **country**.
* **genres** – contains information about the **genres**.
* **movies additional info** – contains information about the **customers**.
* movies\_actors – a **many** to **many** **mapping** table between the **movies** and the **actors**.
* genres\_movies – a **many** to **many** **mapping** table between the **genres** and the **movies**.

## 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 **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| continent | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| currency | A **string** containing a maximum of **5 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |

**genres**

|  |  |  |
| --- | --- | --- |
| **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 **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |

**actors**

|  |  |  |
| --- | --- | --- |
| **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 **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| last\_name | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| birthdate | The birthdate **date** of the person. | **NULL** is **NOT** permitted**.** |
| height | **Integer,** from **1** to **2,147,483,647.** |  |
| awards | **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**.** |

**movies\_additional\_info**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| rating | **DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |
| runtime | **Integer,** from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted**.** |
| picture\_url | A **string** containing a maximum of **80 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| budget | **DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**. |  |
| release\_date | The release **date** of the movie. | **NULL** is **NOT** permitted**.** |
| has\_subtitles | Can be true or false |  |
| description | A very **long** string field |  |

**movies**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| title | A **string** containing a maximum of **70 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |
| country\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table countries.  **NULL** is **NOT** permitted**.** |
| movie\_info\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table movies\_additional\_info.  **NULL** is **NOT** permitted**.**  **UNIQUE** values. |

**movies\_actors**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| movie\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table movies. |
| actor\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table actors. |

**genres\_movies**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| genre\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table genres. |
| movie\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table movies. |

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.

Select and join only tables and columns that are needed in the exercises. Any additional or less information will be considered wrong.

### Insert

You will have to **insert** records of data into the **actors** table.

The new data will be based on **actors** with **id** equal or less than **10**. **Insert data** in the **actors** table with the **following values**:

• **first\_name** – set it to the first name of the actor but **reversed**.

• **last\_name** – set it to the last name of the actor but **reversed**.

• **birthdate** – set it to the **birthdate** of the **actor** but **2 days earlier**.

• **height** – set it to the **height** of the **actor** plus **10**.

• **awards** – set them to the **country\_id**.

• **country\_id** – set it to the **id** of **Armenia**.

### Update

**Reduce** all **movies** **runtime** by 10 minutes for **movies** with **movies\_additional\_info** **id** equal to or greater than 15 and less than 25 (**inclusive**).

### Delete

**Delete** all **countries** that don’t have movies.

## 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.

### Countries

Extract from the **softuni\_imdb** system database, info about the name of **countries**.

**Order** the results by **currency** in **descending** order and then by **id**.

#### Required Columns

* id (countries)
* name
* continent
* currency

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **continent** | **currency** |
| 42 | South Africa | Africa | ZAR |
| 53 | Uzbekistan | Asia | UZS |
| 50 | Uruguay | South America | UYU |
| ... | ... |  |  |

### Old movies

Write a query that returns: **title**, **runtime**, **budget** and **release\_date** from table **movies\_additional\_info**. **Filter** movies which have been **released** from 1996 to 1999 **year** (**inclusive**).

**Order** the results **ascending** by **runtime** then by **id** and show only the first **20** results.

#### Required Columns

* id
* title
* runtime
* budget
* release\_date

#### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **title** | **runtime** | **budget** | **release\_date** |
| 251 | Maniac | 60 | 110495.27 | 1999-10-28 |
| 298 | Ronin | 60 | 447741.91 | 1997-07-25 |
| 103 | Opfergang | 62 | 481899.08 | 1999-09-02 |
| … | … | … | … | … |

### Movie casting

Some actors are free and can apply the casting for a new movie. You must search for them and prepare their documents.

Write a query that returns: **full name**, **email**, **age** and **height** for all actors that are not participating in a movie.

To find their **email** you must take their last name **reversed** followed by the **number of characters** of their last name and then the casting email **“@cast.com”**

**Order** by height in **ascending** order.

#### Required Columns

* full\_name (first\_name + " " + last\_name)
* email (last\_name reversed + number of characters from the last\_name + @cast.com)
* age (2022 – the year of the birth)
* height

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **full\_name** | **email** | **age** | **height** |
| Hube Miranda | adnariM7@cast.com | 35 | 155 |
| Charlotte Eyres | seryE5@cast.com | 55 | 156 |
| … | … | … | … |
| Connie Mackneis | sienkcaM8@cast.com | 27 | 184 |

### International festival

The international movie festival is about to begin. We need to find the countries which are nominated to host the event.

Extract from the database, the **name the country** and the **number of movies** created in this country. The **number of movies** must be higher or equal to **7**.

**Order** the results **descending** by **name.**

#### Required Columns

* name (country)
* movies\_count (number of movies created in the country)

#### Examples

|  |  |
| --- | --- |
| **name** | **movies\_count** |
| Sweden | 13 |
| Serbia | 8 |
| Philippines | 9 |
| … | … |
| Argentina | 7 |

### Rating system

From the database extract the **title**, **rating**, **subtitles,** and the **budget** of **movies**. If the **rating** is equal or less than **4** the user must see “**poor**”, above **4** and less or equal to **7 “good”** and above that it should display “**excellent**”. If the movie has subtitles the user should see “english”, otherwise “-“.

**Order** the results **descending** by **budget**.

#### Required Columns

* title
* rating (less or equal to 4 – “poor”, above 4 and less or equal to 7 – “good”, above 7 – “excellent”)
* subtitles (if it has subtitles it– “english”, otherwise – “-“)
* budget

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **title** | **rating** | **subtitles** | **budget** |
| Metsän tarina | good | english | 499981.78 |
| Family Secrets (Familjehemligheter) | poor | - | 497338.13 |
| Place in the Sun, A (En plats i solen) | excellent | english | 496586.35 |
| . . . | . . . | . . . | . . . |
| Saban, Son of Saban | good | - | 21027.33 |

## 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.

### History movies

Create a **user defined function** with the name **udf\_actor\_history\_movies\_count(full\_name VARCHAR(50))** that receives an **actor’s full name** and returns the total number of **history** movies in which the actor has a role.

#### Required Columns

* history\_movies(udf\_customer\_products\_count)

#### Example

|  |
| --- |
| **Query** |
| SELECT udf\_actor\_history\_movies\_count('Stephan Lundberg') AS 'history\_movies'; |
| history\_movies |
| **2** |

|  |
| --- |
| **Query** |
| SELECT udf\_actor\_history\_movies\_count('Jared Di Batista') AS 'history\_movies'; |
| history\_movies |
| **1** |

### Movie awards

A movie has won an award. Your task is to find all actors and give them the award.

Create a stored procedure udp\_award\_movie which accepts the following parameters:

* movie\_title(VARCHAR(50))

Extracts data about the **movie** with the given **title** and find all **actors** that play in it and **increase** their **awards** with **1**.

#### Result

|  |
| --- |
| **Query** |
| CALL udp\_award\_movie('Tea For Two'); |
| This execution will update 3 actors – Vanna Bilborough, Armando Cabrera, Ingrid Ackenhead |
| Result |
| |  |  |  |  | | --- | --- | --- | --- | | **full\_name** | **awards**  before | **->** | **awards**  after | | Vanna Bilborough | 20 | -> | 21 | | Armando Cabrera | 18 | -> | 19 | | Ingrid Ackenhead | 24 | -> | 25 | |