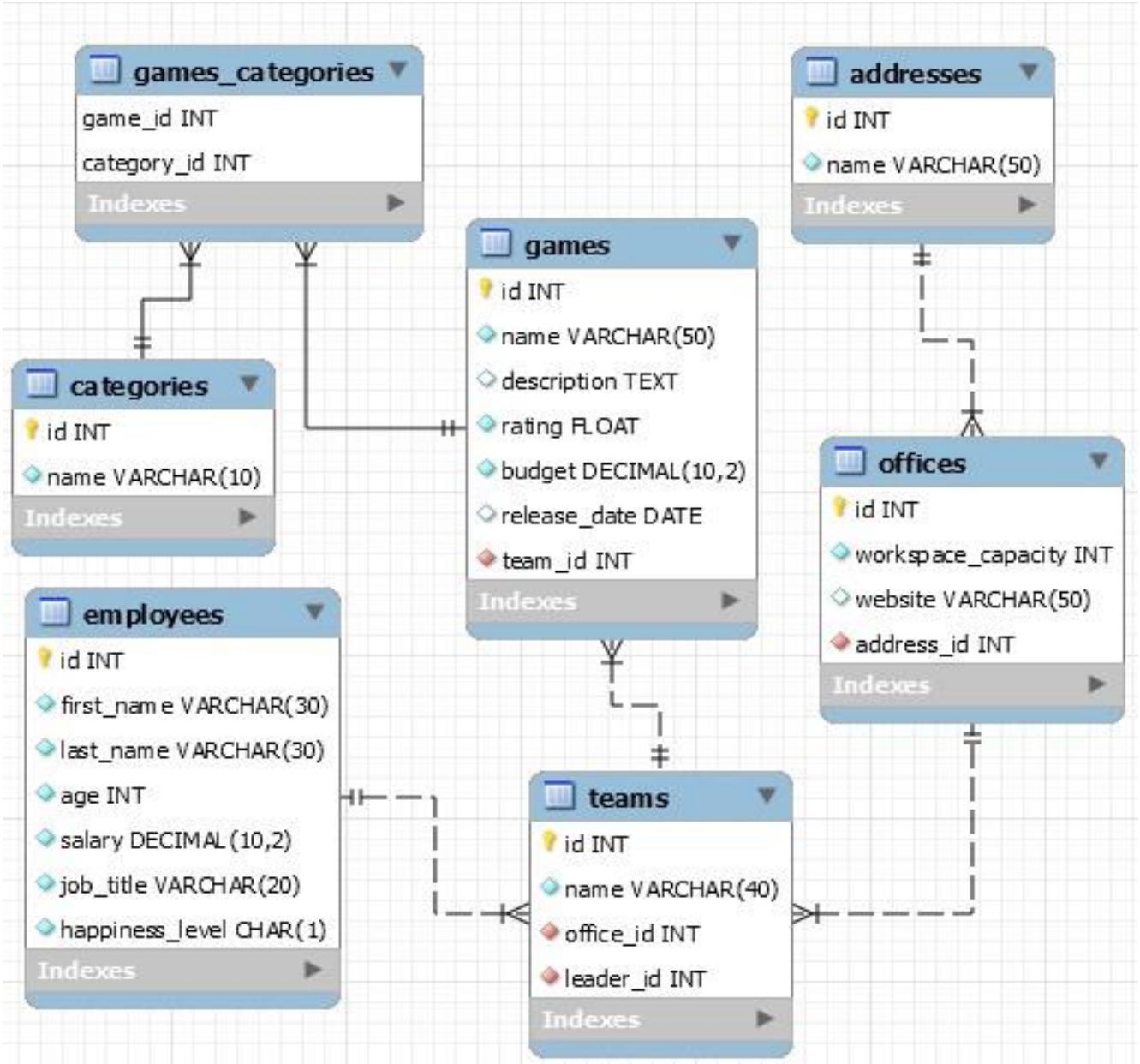
MySQL Retake Exam
SoftUni Game Dev Branch

Quite normally, the majority of SoftUni employees are avid fans of computer games. And this does not go unnoticed by their managers. They immediately decided that it was the right time for SoftUni to enter the ever-growing computer game market. And because they need a database, managers turn to you for help. You will receive a document with detailed explanations of what this database should be. You will also receive test data, with which you will be able to do many tests to prove that you have managed to fully cope with your task.

Section 0: Database Overview

You have been given an Entity / Relationship Diagram of the SoftUni Game Dev Branch:



The **SoftUni Game Dev Branch (sgd)** needs to hold information about **games**, **teams**, **employees**, **offices, addresses and categories**.

Your task is to create a database called **sgd (SoftUni Game Dev Branch)**.
Then you will have to create several **tables**.

- **games** – contains the information about the **games**.
  - Each game has a name, a description, a rating, a budget, a release date and a team.
- **teams** – contains the information about the **teams**.
  - The team has a name, a leader, and an office.
- **employees** – contains the information about the **employees**
  - Each employee has a first and last name, an age, a salary, a job title and a happiness level.
- **offices** – contains the information about the **offices**.
  - The office has a workspace capacity, a website, and an address
- **addresses** – contains the information about the **addresses**.
  - The address name.
- **categories** – contains the information about the **categories**.
  - The category name.
- **games_categories** – a **many** to **many mapping** table between the **games** and the **categories**.
  - Has a **composite primary** key from the game_id and the category_id

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 will be given will be the minimal required for you to implement the database.

1. Table Design

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

addresses

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**.**

categories

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 **10 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**

`offices`

Column Name	Data Type	Constraints
`id`	**Integer**, from **1** to **2,147,483,647.**	**Primary Key AUTO_INCREMENT**
`workspace_capacity`	**Integer**, from **1** to **2,147,483,647.**	**NULL** is **NOT** permitted**.**
`website`	A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed.	**NULL is permitted.**
`address_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with `addresses`. **NULL** is **NOT** permitted**.**

`employees`

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 **30 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**
`last_name`	A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**
`age`	**Integer,** from **1** to **2,147,483,647.**	**NULL** is **NOT** permitted**.**
`salary`	**DECIMAL number**, up to **10 digits**, **2** of which after the **decimal point**.	**NULL** is **NOT** permitted**.**
`job_title`	A **string** containing a maximum of **20 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**
`happiness_level`	**A character** that shows the **happiness level** of the employee. Can be 'L'- Low, 'N' - Normal or 'H'- High	**NULL** is **NOT** permitted**.**

`teams`

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**.**
`office_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table `offices`. **NULL** is **NOT** permitted**.**
`leader_id`	**Integer**, from **1** to **2,147,483,647.**	Relationship with table `employees`. **The values are UNIQUE.**

		NULL is **NOT** permitted**.**

games

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**
description	A very **long** string field.	**NULL is permitted**.
rating	A **floating** point **numbers**.	**DEFAULT value** is **5.5** **NULL** is **NOT** permitted
budget	**DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**.	**NULL** is **NOT** permitted**.**
release_date	The release **date** of the game.	**NULL is permitted**.
team_id	**Integer**, from **1** to **2,147,483,647.**	Relationship with table **teams**. **NULL** is **NOT** permitted**.**

games_categories

Column Name	Data Type	Constraints
game_id	**Integer**, from **1** to **2,147,483,647.**	**NULL** is **NOT** permitted.
category_id	**Integer**, from **1** to **2,147,483,647.**	**NULL** is **NOT** permitted.

- The **games_categories** table has a **composite primary key** from game_id and category_id.

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 data which you will need to **store** in your **local database**. This data will be given to you, so you do not have to imagine it and lose precious 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.

2. Insert

The bosses urgently want to announce 9 new games and because there is no time, the developers decide not to waste time thinking about details but to announce something as soon as possible.

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

For all **teams** with **id between 1** and **9** (**both inclusive**), **insert data** in the **games** table with the **following values**:

- **name**:
  - the name of the team but **reversed**

    - all letters must be **lower** case
    - **omit** the **starting** character of the team's name
        - Example: Team name – **T**hiel -> leih
- **`rating`** – set it to be equal to the team's id
- **`budget`** – set it to be equal to the leader's id multiplied by 1000
- **`team_id`** –-set it to be equal to the team's id

3. Update

After a good work in recent months, management has decided to raise the salaries of all young team leaders.

Update all **young employees (only team leaders) with age under 40(exclusive)** and **increase** their **salary** with **1000**.

Skip the employees with salary **over 5000(inclusive)**. Their salaries are already high.

4. Delete

After a lot of manipulations on our base, now we must clean up.

Delete all **`games`** from table **`games`,** which do not have a **category** and **release date.**

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 have been given, to ensure **maximum consistency** with the **examples** given in this section.

5. Employees

Extract from the **SoftUni Game Dev Branch (sgd)** database, info about all the **employees**.

Order the results by the **employee's salary,** then by their **id**.

Required Columns

- **`first_name`**
- **`last_name`**
- **`age`**
- **`salary`**
- **`happiness_level`**

Example

first_name	last_name	age	salary	happiness_level
Bondon	Toquet	20	1289.90	H
Eldon	Dot	46	1321.13	N
Garrett	Jocelyn	30	1378.71	L
…	…	…	…	…
Baron	Sange	20	8866.37	H

Roley	Robertz	45	8987.87	L

6. Addresses of the teams

Extract from the database all the **team names** and their **addresses**. Also display the **count of the characters of the address names.**

Skip those teams whose office **does not** have a website.

Order the results by **team names,** then by the **address names**.

Required Columns

- **`team_name`**
- **`address_name`**
- **`count_of_characters(of the address name)`**

Example

team_name	address_name	count_of_characters
Abbott, Deckow and Goyette	20605 Helena Lane	17
Armstrong	49099 Manitowish Court	22
Bartoletti-King	10 Jenna Park	13
…	…	…
Yost Group	31314 Butterfield Lane	22
Yundt	444 Golden Leaf Place	21

7. Categories Info

Now, we need a more detailed information about categories – count of game, average budget and max rating.

Select all **categories names**, **count** of the **games** from each category, the **average budget (rounded to the** second **digit after the decimal point)** of all games from the current category and the **max rating** of games from a category.

Order the result by **count of games** in **descending** order, then by the **name** of the category alphabetically.

Skip categories with **max rating** lower than **9.5**(exclusive).

Required Columns

- **`name`**
- **`games_count`**
- **`avg_budget (rounded to the second digit after the decimal point)`**
- **`max_rating`**

Example

name	games_count	avg_budget	max_rating

Puzzle	18	54340.62	**9.8**
Action	14	46425.07	**9.6**
MMORPG	14	57006.74	**9.5**
Strategy	14	39754.41	**9.6**
Sports	13	41122.07	**9.8**

8. Games of 2022

Now, we need to find all interesting upcoming games.

Extract from the database all games that are being **released in the year 2022**. Also, the **month** must be **even**. We need only the first game **sequel** (ends with '…2'). We need the information of the **game name**, the game **release date**, a short **summary** (only the first 10 characters + '…') and the name of the team.

At last, a column '**Quarters'** depends on the month of the release date:

- January, February, and March (Q1)
- April, May, and June (Q2)
- July, August, and September (Q3)
- October, November, and December (Q4)

Order by **Quarters**.

Required Columns

- **name** (of the game)
    - **only the first sequel**
        - Ends with '...2'
        - Voyatouch 2 -> Valid
        - Voyatouch 3 -> Invalid
- **release_date**
    - only even months
- **summary**
    - the first 10 characters + '...'
- **quarter**
    - Depends on the month
- **team_name** (name of the team)

Example

name	release_date	summary	quarter	team_name
Y-Solowarm 2	2022-02-28	In hac hab...	Q1	Jenkins-Kiehn

Mat Lam Tam 2	2022-08-21	Proin leo ...	Q3	Roob, Mann and Goldner
Voyatouch 2	2022-12-26	In sagitti...	Q4	Weissnat-Wolf

9. Full info for games

Our managers want to monitor all games that **don't have** a **release date nor** a **category**. They want us to create a query, which shows the main information about the games. The information that they need is the **name** of the **game**, the **name** of the **team**, the **name** of the **address** and if the budget is **less than 50000**. If it is, we need to display **'Normal budget'**. If it doesn't - **'Insufficient budget'**.
Finally, we should order the result by the **name** of the **game**.

Required Columns

- **`name (of the game)`**
- **`budget_level`**
- **`team_name`**
- **`address_name`**

Example

name	budget_level	team_name	address_name
Bitwolf 2	Normal budget	Flatley Group	49099 Manitowish Court
Lotlux	Insufficient budget	Boyer, Stamm and Schinner	263 Glendale Lane
Mat Lam Tam 3	Normal budget	McLaughlin	88229 Norway Maple Court
Regrant	Normal budget	O'Kon-Mosciski	3569 Canary Lane
Stringtough 2	Insufficient budget	Hoeger Group	49099 Manitowish Court
Tin 2	Insufficient budget	Dibbert	5 Sunbrook Point

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.

10. Find all basic information for a game

Create a **user defined function** with the name **`udf_game_info_by_name (game_name VARCHAR (20))`** that receives a **game's name** and returns the basic information as a text sentence.

- Example
    - `The "game_name" is developed by a "team_name" in an office with an address "address_text"`

Example 1

Query
`SELECT udf_game_info_by_name('Bitwolf') AS info;`
info
The **Bitwolf** is developed by a **Rempel-O'Kon** in an office with an address **92 Memorial Park**

Example 2

Query
`SELECT udf_game_info_by_name('Fix San') AS info;`
info
The **Fix San** is developed by a **Schulist** in an office with an address **75 Harper Way**

Example 3

Query
`SELECT udf_game_info_by_name('Job') AS info;`
info
The **Job** is developed by a **Shields Group** in an office with an address **036 Stuart Pass**

11. Update budget of the games

We will have to increase the support of the games that do not have any categories yet. We should find them and increase their budget, as well as push their release date

The procedure must **increase** the **budget** by **100,000** and **add one year** to their **release_date** to the games that do **not have any categories** and their **rating** is **more than (not equal)** the given parameter **min_game_rating** and **release date is NOT NULL.**

Create a stored procedure **udp_update_budget** which accepts the following parameters:

- **min_game_rating(floating point number)**

Query
`CALL udp_update_budget (8);`
This execution will update three games – **Quo Lux, Daltfresh** and **Span.**
Result
Quo Lux - 23384.32 -> **123384.32** | 2022-06-26 -> **2023-06-26** Daltfresh - 86012.38 -> **186012.38** | 2021-06-17 -> **2022-06-17** Span - 47468.36 -> **147468.36** | 2022-06-05 -> **2023-06-05**