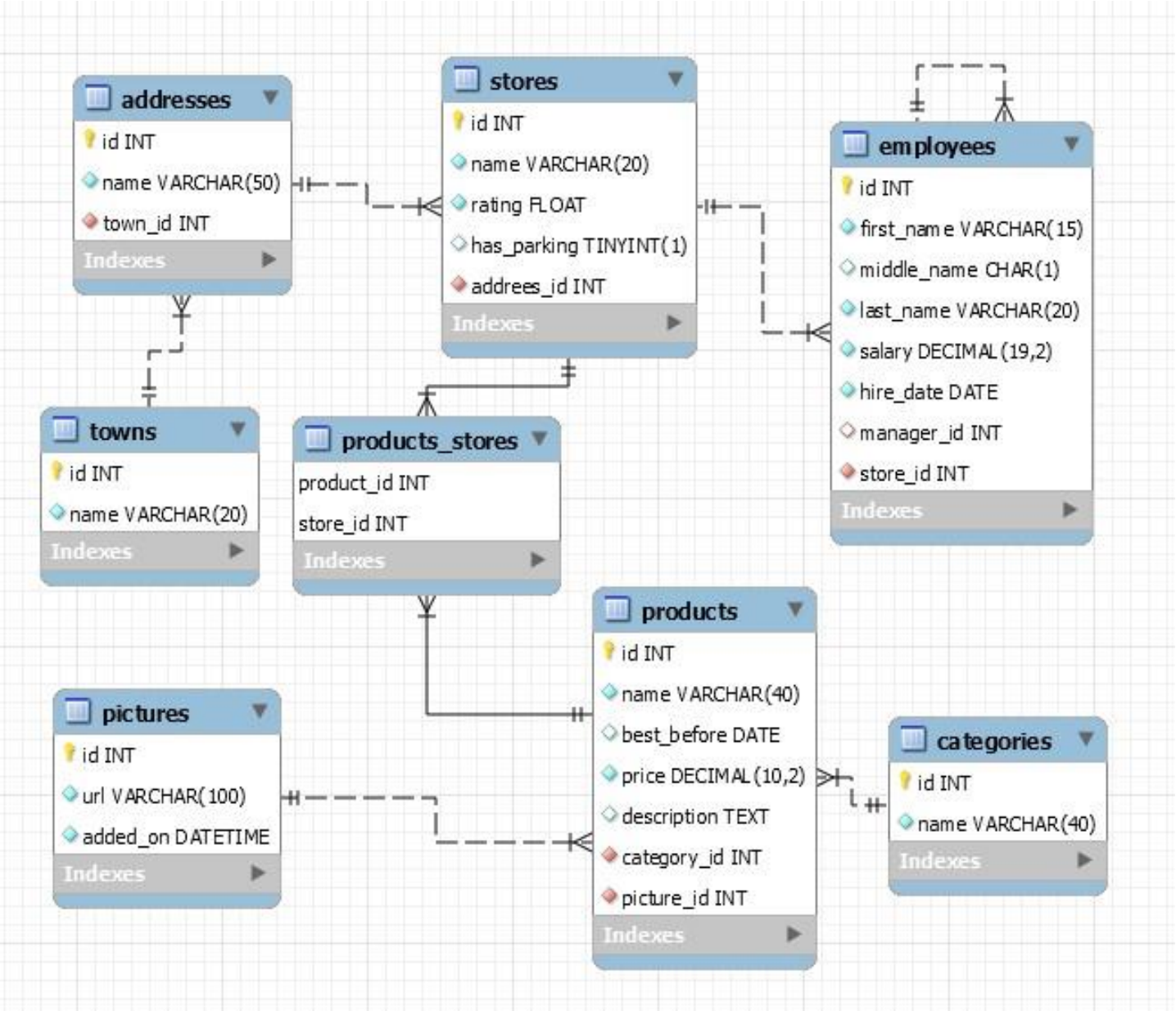
MySQL Exam

Triple S – SoftUni Stores System

Because of the fact that the students in the Java Track are the best in SoftUni, with a look into the future, they decided to create databases for all eventually future businesses of the SoftUni. Of course, they have many ideas, but they need to start from somewhere. You have more than year experience, that's why you were chosen for a senior developer for one of the teams. Your task is to create a store system – SoftUni Stores System. You and the other senior developers create an E/R Diagram, that looks like this. Good Luck.

Section 0: Database Overview

You have been given an Entity / Relationship Diagram of the SoftUni Stores System:



The **SoftUniStoresSystem** needs to hold information about **stores**, **products**, **employees**, **addresses, towns, pictures** and **categories**.

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

- **stores** – contains information about the **stores.**
  - Each store has a **name, rating, has parking** and relation with **addresses.**

- **products** – contains information about the **products.**
  - Each product has a **name, best before, price, description** and has relations with **categories** and **pictures.**

- **products_stores** – a **many** to **many mapping** table between the **products** and the **stores.**
  - Has a **composite primary key** from product_id and store_id

- **employees** – contains information about the **employees.**
  - Each employee has **first name, middle name, last name, salary** and have **relations** with **stores** and with **self.**

- **addresses** – contains information about the **addresses** of stores.
  - Each address has **name** and **relation** with **towns.**

- **towns** - contains information about the **towns.**
  - Each town has a **name.**

- **categories** – contains information about the **categories.**
  - Each category has a **name.**

- **pictures** – contains information about the **pictures.**
  - Each picture has a **name** and **date and time** when is added on.

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 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:

`pictures`

Column Name	Data Type	Constraints
`id`	**Integer,** from **1** to 2,147,483,647.	**Primary Key** **AUTO_INCREMENT**
`url`	A **string** containing a maximum of **100 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**
`added_on`	A **date** and **time** of adding picture.	**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 **40 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.** The name is **unique**.

`products`

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**.** The name is **unique**.
`best_before`	A **date** that product is best before	
`price`	**Decimal number**, up to **10 digits**, **2** of which after the **decimal point**.	**NULL** is **NOT** permitted**.**
`description`	A **very long** String field	
`category_id`	**Integer**, from **1** to 2,147,483,647.	Relationship with table `categories`. **NULL** is **NOT** permitted**.**
`picture_id`	**Integer**, from **1** to 2,147,483,647.	Relationship with table `pictures`. **NULL** is **NOT** permitted**.**

`towns`

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 **20 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.** The name is **unique**.

`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**.** The name is **unique**.
`town_id`	**Integer**, from **1** to 2,147,483,647.	Relationship with table **`towns`**. **NULL** is **NOT** permitted**.**

`stores`

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 **20 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.** The name is **unique**.
`rating`	A floating point number	**NULL** is **NOT** permitted.
`has_parking`	Can be true or false	**Default** is **FALSE**
`address_id`	**Integer**, from **1** to 2,147,483,647.	Relationship with table **`addresses`**. **NULL** is **NOT** permitted**.**

`products_stores`

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

`store_id`	**Integer**, from **1** to 2,147,483,647.	**NULL** is **NOT** permitted.

- `products_stores table has a composite primary key from product_id and store_id`

`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 **15 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**
`middle_name`	A single one character	
`last_name`	A **string** containing a maximum of **20 characters**. Unicode is **NOT** needed.	**NULL** is **NOT** permitted**.**
`salary`	**Decimal number**, up to **19 digits**, **2** of which after the **decimal point**.	**DEFAULT 0**
`hire_date`	A **date** that employee was **hired**	**NULL** is **NOT** permitted**.**
`manager_id`	**Integer**, from **1** to 2,147,483,647.	
`store_id`	**Integer**, from **1** to 2,147,483,647.	**NULL** is **NOT** permitted**.**

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 don't 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

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

Find all **products** that are **not offered** in any stores (don't have a relation with stores) and insert data in the products_stores. For every product saved -> **product_id** and **1(one)** as a **store_id**. And now this product will be offered in store with name **Wrapsafe** and **id 1**.

- `product_id` – id of product
- `store_id` – set it to be 1 for all products.

3. Update

Update all **employees** that hire **after 2003(exclusive)** year and **not work** in store **Cardguard** and **Veribet**.
Set their **manager** to be **Carolyn Q Dyett** (with **id 3**) and **decrease salary** with 500.

4. Delete

It is time for the stores to start working. All good employees already are in their stores. But some of the employers are too expensive and we need to cut them, because of finances restrictions.
Be careful not to delete **managers they are also employees**.
Delete only those employees that **have managers** and a salary is more than **6000**(inclusive)

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.

5. Employees

Extract from the SoftUni Stores System database, info about all of the **employees**.

Order the results by employees **hire date** in **descending** order.

Required Columns

- `first_name`
- `middle_name`
- `last_name`
- `salary`
- `hire_date`

Example

first_name	middle_name	last_name	salary	hire_date
Roz	U	Dewdney	9316.56	2018-10-20
Florian	E	Bamlet	6266.27	2018-02-19
Shae	O	Fasey	7463.52	2018-02-03
Elwin	G	Rennock	9538.20	2017-05-12
…	…	…	…	…

Carolyn	Q	Dyett	1223.45	2000-02-23

6. Products with old pictures

A photographer wants to take pictures of **products that have old pictures**. You must select all of the products that have a description **more than 100 characters long description**, and a **picture that is made before 2019 (exclusive)** and the product **price** being **more** than **20**. Select a **short description** column that consists of **first 10 characters** of the picture's description **plus '...'**. Order the results by product **price** in **descending** order.

Required Columns

- `name (product)`
- `price`
- `best_before`
- `short_description`
  - `only first 10 characters of product description + '...'`
- `url`

Example

product_name	price	best_before	short_description	url
Pasta - Bauletti, Chicken White	48.85	2020-02-08	Fusce cong...	http://dummyimage.com/241x194.jpg/5fa2dd/ffffff
Oil - Sunflower	48.00	2019-10-25	Lorem ipsu...	http://dummyimage.com/243x233.jpg/cc0000/ffffff
Sugar - White Packet	40.89	2019-11-14	Pellentesq...	http://dummyimage.com/197x104.jpg/cc0000/ffffff
...				
Lemonade - Mandarin, 591 Ml	25.53	2020-04-03	Duis biben...	http://dummyimage.com/208x226.jpg/cc0000/ffffff

7. Counts of products in stores and their average

The managers needs to know in which stores sell different products and their average price.

Extract from the database all of the **stores (with** or **without** products**)** and the **count** of the **products** that they have. Also you can show the average price of all products (rounded to the second digit after decimal point) that sells in store.

Order the results **descending by count of products in store**, then by **average** price in **descending order** and finally by **store id**.

Required Columns

- `Name (store)`
- `product_count`
- `avg`

Example

name	product_count	avg
DuoStore	4	32.15
Home Ing	3	13.72
Alphazap	2	48.43
Duobam	2	44.45
…	…	…
Lotstring	0	NULL

8. Specific employee

There are many employees in our shop system, but we need to find only the one that passes some specific criteria.

Extract from the database, the **full name** of employee, **name** of **store** that he works, **address** of store, and **salary**. The employee's **salary** must be **lower** than **4000**, the **address** of the store must **contain** '5' somewhere, the **length** of the **store name** needs to be **more than** 8 characters and the employee's **last name** must **end** with an 'n'.

Required Columns

- `Full name (employee)`
- `Store name`
- `Address`
- `Salary`

Example

Full_name	Store_name	address	salary
Leigh Vedenyakin	Stronghold	32759 Dwight Plaza	2159.55

9. Find all information of stores

The managers always want to know how the business goes. Now, they want from us to show all store names, but for security, the name and must be in the reversed order.

Select the name of stores (in **reverse** order).

After that, the full_address in format: {**town name** in **upper case**}-{**address** name}.

The next info is the **count** of **employees**, that work in the store.

Filter only the stores that have a **more than one** employee.

Order the results by the **full_address in ascending order.**

Required Columns

- `reversed_name (store name)`
- `full_address (full_address)`
- `employees_count`

Example

reversed_name	full_address	employees_count
dlohgnortS	BLAGOEVGRAD-32759 Dwight Plaza	3
mabouD	BLAGOEVGRAD-35952 Stoughton Circle	1
focsnarT	BURGAS-07 Armistice Parkway	2
…	…	…
draugdraC	VIDIN-61346 Melody Lane	3

Section 4: Programmability – 30 pts

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

10. Find full name of top paid employee by store name

Create a **user defined function** with the name **`udf_top_paid_employee_by_store(store_name VARCHAR(50))`** that receives a **store name** and returns the **full name** of **top paid employee**.
Full info must be in format:

{**first_name**} {**middle_name**}**.** {**last_name**} works in store for {**years of experience**} years

The years of experience is the difference when **they were hired and 2020-10-18**

Example 1

Query
`SELECT udf_top_paid_employee_by_store('Stronghold') as 'full_info';`
full_info
Breena S. Hymans works in store for **3** years

Example 2

Query
`SELECT udf_top_paid_employee_by_store('Keylex') as 'full_info';`
full_info
Xylina W. Apfelmann works in store for **7** years

11. Update product price by address

CREATE user define **procedure udp_update_product_price (address_name VARCHAR (50)),** that receives as parameter an **address name**.
Increase the product's price with **100** if the address **starts with 0 (zero)** otherwise **increase** the price with **200**.

Example 1

Query
`CALL udp_update_product_price('07 Armistice Parkway');` `SELECT name, price FROM products WHERE id = 15;`

Result

name	price
Spic And Span All Purpose	136.53

Example 2

Query
`CALL udp_update_product_price('1 Cody Pass');` `SELECT name, price FROM products WHERE id = 17;`

Result

name	price
Wine - Ruffino Chianti Classico	221.63