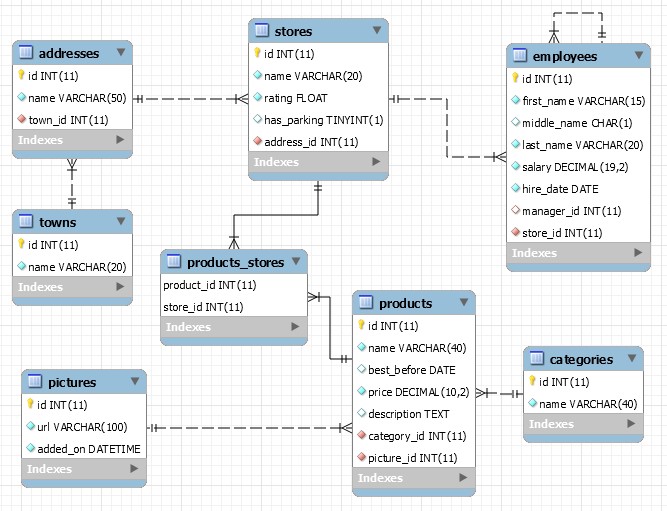
# MySQL Exam

# Triple S – SoftUni Stores System

## Because of the fact that the students in the Java Track are the best in SoftUni, with а 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.

### 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. |

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 output to the Judge system from the following sql queries:**

### Table Design – 13pts

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_SCHEMA = 'softuni\_stores\_system'

ORDER BY COLUMN\_NAME, TABLE\_NAME;

| **output** |
| --- |
| added\_on |
| address\_id |
| best\_before |
| … |

### Table Design – 13pts

SELECT COLUMN\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_SCHEMA = 'softuni\_stores\_system'

ORDER BY COLUMN\_NAME, TABLE\_NAME;

| **output** |
| --- |
| datetime |
| int |
| date |
| … |

**NOTE**: If your output displays **column type int** as **int(11)**, you need to remove the **(11)**  
**int(11) != int**

### Table Design – 14pts

SELECT COLUMN\_KEY FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_SCHEMA = 'softuni\_stores\_system'

AND COLUMN\_NAME IN ('id','product\_id','store\_id',

'town\_id', 'address\_id', 'category\_id', 'picture\_id','manager\_id')

ORDER BY COLUMN\_NAME, TABLE\_NAME DESC, COLUMN\_KEY DESC;

| **output** |
| --- |
| MUL |
| MUL |
| PRI |
| … |

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

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

**Execute the following SQL statements and paste the result in judge as a solution**

### 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.

SELECT store\_id, s.name, p.name, product\_id FROM products\_stores

JOIN products p ON p.id = products\_stores.product\_id

JOIN stores s ON products\_stores.store\_id = s.id

ORDER BY product\_id, store\_id;

| **store\_id** | **store\_name** | **product\_name** | **product\_id** |
| --- | --- | --- | --- |
| 11 | Home Ing | Container Clear 8 Oz | 1 |
| 14 | Veribet | Container Clear 8 Oz | 1 |
| 18 | DuoStore | Container Clear 8 Oz | 1 |
| … | … | … | ... |
| 6 | Duobam | Sugar - White Packet | 20 |

### 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.

SELECT first\_name, salary, hire\_date, id

FROM employees

WHERE manager\_id = 3;

| **first\_name** | **salary** | **hire\_date** | **id** |
| --- | --- | --- | --- |
| Clemence | 8753.81 | 2008-10-12 | 1 |
| Fletch | 1644.71 | 2007-04-14 | 5 |
| Roz | 8816.56 | 2018-10-20 | 6 |
| … | … | … | ... |
| Frasquito | 8624.43 | 2011-11-25 | 19 |

### 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)

SELECT first\_name, salary, hire\_date, id

FROM employees;

| **first\_name** | **salary** | **hire\_date** | **id** |
| --- | --- | --- | --- |
| Florian | 6266.27 | 2018-02-19 | 2 |
| Carolyn | 1223.45 | 2000-02-23 | 3 |
| Eba | 5268.40 | 2015-02-09 | 4 |
| … | … | … | ... |
| Leigh | 2159.55 | 2002-07-11 | 20 |

## 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.

### 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 |

### 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 |

### Count 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 |

### Specific employees

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

Extract from the database, the **full name** (a combination of **first and last** name) of employee, **name** of **store** that he works, **address** of store, and **salary**. The employee's **salary** must be **lower** than **7000**, the **address** of the store must **contain** 'a' somewhere and the **length** of the **store name** needs to be **more than** 5 characters. Order the result by **employee id**.

#### Required Columns

* Full name (employee)
* Store name
* Address
* Salary

#### Example

| Full\_name | Store\_name | address | **salary** |
| --- | --- | --- | --- |
| Florian Bamlet | Veribet | 29395 Larry Pass | 6266.27 |
| Carolyn Dyett | Cardguard | 61346 Melody Lane | 1223.45 |
| ... | ... | ... | ... |
| Leigh Vedenyakin | Stronghold | 32759 Dwight Plaza | 2159.55 |

### Find all information about 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**, which work in the store, the **minimum** **price** of products that are sold in the store and the **count of product ids**.

**Filter** only the stores that have a **minimum** **price** of product **more** than 10.

The last one is the date of the **newest** **added picture** of product, which is sold in the store, in a specific format: 4th-Dec-2018 (Example, Day of the month **should be with English suffix** (1st, 2nd, 3rd …))

* Note that 4th-Dec-2018 is different from 4-Dec-2018 and 2018-12-4

**Order the results** by the **reversed name** of the store in ascending order, then by the **minimum price** of product in stores

#### Required Columns

* reversed\_name (store name)
* full\_address (full\_address)
* employees\_count
* min\_price
* products\_count
* newest\_pic – in a specific format

#### Example

| **reversed\_name** | full\_address | employees\_count | min\_price | products\_count | newest\_pic |
| --- | --- | --- | --- | --- | --- |
| balknoK | BLAGOEVGRAD-789 Lien Plaza | 0 | 29.95 | 2 | 12th-Jun-2020 |
| dlohgnortS | BLAGOEVGRAD-32759 Dwight Plaza | 3 | 25.53 | 1 | 21st-Aug-2018 |
| draugdraC | VIDIN-61346 Melody Lane | 3 | 23.21 | 1 | 14th-Nov-2018 |
| ezeerbraloS | DOBRICH-1 Cody Pass | 0 | 21.63 | 2 | 6th-Jan-2020 |
| … | … | … | … | … | … |
| ydixelF | VIDIN-83637 Reinke Alley | 0 | 25.53 | 1 | 21st-Aug-2018 |

## 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.

*ATTENTION: The solution for the next problems will be evaluated MANUALLY. PLEASE IGNORE THE RESULTS IN JUDGE.*

### Find full name of top paid employee by store

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 |

### 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** |