Exercises: Data Definition and Data Types

This document defines the **exercise assignments** for the "Databases Basics - MySQL" course @ Software University.

Problem 1. Create Database

You now know how to create database using the GUI of the HeidiSQL. Now it's time to create it using SQL queries. In that task (and the several following it) you will be required to create the database from the previous exercise **using only SQL queries**. Firstly, just **create new database** named **minions.**

Problem 2. Create Tables

In the newly created database Minions add table **minions (id, name, age)**. Then add new table **towns (id, name).** Set **id** columns of both tables to be **primary key** as **constraint**.

Problem 3. Alter Minions Table

Change the structure of the Minions table to have **new column town_id** that would be of the same type as the **id** column of **towns table**. Add **new constraint** that makes **town_id foreign key** and references to **id** column of **towns** table.

Problem 4. Insert Records in Both Tables

Populate both tables with sample records given in the table below.

minions			
id	**name**	**age**	**town_id**
1	Kevin	22	1
2	Bob	15	3
3	Steward	NULL	2

towns	
id	**name**
1	Sofia
2	Plovdiv
3	Varna

Use only SQL queries. Submit your **INSERT statements** in Judge as Run skeleton, run queries & check DB.

Problem 5. Truncate Table Minions

Delete all the data from the **minions** table using **SQL query.**

Problem 6. Drop All Tables

Delete all tables from the **minions** database using **SQL query**.

Problem 7. Create Table People

Using **SQL query** create table Users with columns:

- **id** – unique number for every person there will be **no more than 2^{31-1} people.** (Auto incremented)
- **name** – full name of the person will be **no more than 200 Unicode characters**. (Not null)
- **picture** – image with **size up to 2 MB.** (Allow nulls)
- **height** – In meters. Real number precise up to **2 digits** after floating point. (Allow nulls)
- **weight** – In kilograms. Real number precise up to **2 digits** after floating point. (Allow nulls)
- **gender** – Possible states are **m** or **f.** (Not null)

- **birthdate –** (Not null)
- **biography** – detailed biography of the person it can contain **max allowed Unicode characters.** (Allow nulls)

Make **id** primary key. Populate the table with **5 records**. Submit your **CREATE** and **INSERT statements** as Run queries & check DB.

Problem 8. Create Table Users

Using **SQL query** create table **users** with columns:

- **id** – unique number for every user. There will be **no more than 2^{63-1} users.** (Auto incremented)
- **username** – unique identifier of the user will be **no more than 30 characters (non Unicode).** (Required)
- **password** – password will be **no longer than 26 characters (non Unicode).** (Required)
- **profile_picture** – image with **size up to 900 KB.**
- **last_login_time**
- **is_deleted** – shows if the user deleted his/her profile. Possible states are **true** or **false**.

Make **id** primary key. Populate the table with **5 records**. Submit your **CREATE** and **INSERT statements**. Submit your **CREATE** and **INSERT statements** as Run queries & check DB.

Problem 9. Change Primary Key

Using **SQL queries** modify table **users** from the previous task. First **remove current primary key** then create **new primary key** that would be **combination** of fields **id** and **username**.

Problem 10. Add Check Constraint

Using **SQL queries** modify table **users**. Add **check constraint** to ensure that the values in the **password** field are **at least 5 symbols** long.

Problem 11. Set Default Value of a Field

Using **SQL queries** modify table **users**. Make the **default value** of **last_login_time** field to be the **current time.**

Problem 12. Set Unique Field

Using **SQL queries** modify table **users**. Remove **username** field from the primary key so only the field **id** would be primary key. Now **add unique constraint** to the **username** field to ensure that the values there are **at least 3 symbols** long.

Problem 13. Movies Database

Using **SQL queries** create **Movies** database with the following entities:

- **directors** (id, director_name, notes)
- **genres** (id, genre_name, notes)
- **categories** (id, category_name, Notes)
- **movies** (id, title, director_id, copyright_year, length, genre_id, category_id, rating, notes)

Set most **appropriate data types** for each column. **Set primary key** to each table. Populate each table with **5 records**. Make sure the columns that are present in 2 tables would be of the **same data type**. Consider which fields

are always required and which are optional. Submit your **CREATE TABLE** and **INSERT statements** as Run queries & check DB.

Problem 14. Car Rental Database

Using **SQL queries** create **car_rental** database with the following entities:

- **categories** (id, category, daily_rate, weekly_rate, monthly_rate, weekend_rate)
- **cars** (id, plate_number, make, model, car_year, category_id, doors, picture, condition, available)
- **employees** (id, first_name, last_name, title, notes)
- **customers** (id, driver_licence_number, full_name, address, city, zip-code, notes)
- **rental_orders** (id, employee_id, customer_id, car_id, car_condition, tank_level, kilometrage_start, kilometrage_end, total_kilometrage, start_date, end_date, total_days, rate_applied, tax_rate, order_status, notes)

Set most **appropriate data types** for each column. **Set primary key** to each table. Populate each table with **3 records**. Make sure the columns that are present in 2 tables would be of the **same data type**. Consider which fields are always required and which are optional. Submit your **CREATE TABLE** and **INSERT statements** as Run queries & check DB.

Problem 15. Hotel Database

Using **SQL queries** create **Hotel** database with the following entities:

- **employees** (id, first_name, last_name, title, notes)
- **customers** (account_number, first_name, last_name, phone_number, emergency_name, emergency_number, notes)
- **room_status** (room_status, notes)
- **room_types** (room_type, notes)
- **bed_types** (bed_type, notes)
- **rooms** (room_number, room_type, bed_type, rate, room_status, notes)
- **payments** (id, employee_id, payment_date, account_number, first_date_occupied, last_date_occupied, total_days, amount_charged, tax_rate, tax_amount, payment_total, notes)
- **occupancies** (id, employee_id, date_occupied, account_number, room_number, rate_applied, phone_charge, notes)

Set most **appropriate data types** for each column. **Set primary key** to each table. Populate each table with **3 records**. Make sure the columns that are present in 2 tables would be of the **same data type**. Consider which fields are always required and which are optional. Submit your **CREATE TABLE** and **INSERT statements** as Run queries & check DB.

Problem 16. Create SoftUni Database

Now create bigger database called **softuni.** You will use database in the future tasks. It should hold information about

- **towns** (id, name)
- **addresses** (id, address_text, town_id)
- **departments** (id, name)
- **employees** (id, first_name, middle_name, last_name, job_title, department_id, hire_date, salary, address_id)

Id columns are **auto incremented** starting from 1 and increased by 1 (1, 2, 3, 4…). Make sure you **use appropriate data types** for each column. Add **primary** and **foreign keys as constraints** for each table. Use **only SQL queries**. Consider which fields are always required and which are optional.

Problem 17. Backup Database

Backup the database **SoftUni** from the previous tasks into a file named "**softuni-backup.bak**". Delete your database from SQL Server Management Studio. Then restore the database from the created backup.

Problem 18. Basic Insert

Use the **SoftUni** database and insert some data **using SQL queries**.

- **Towns:** Sofia, Plovdiv, Varna, Burgas
- **Departments:** Engineering, Sales, Marketing, Software Development, Quality Assurance
- **Employees:**

Name	Job Title	Department	Hire Date	Salary
Ivan Ivanov Ivanov	.NET Developer	Software Development	01/02/2013	3500.00
Petar Petrov Petrov	Senior Engineer	Engineering	02/03/2004	4000.00
Maria Petrova Ivanova	Intern	Quality Assurance	28/08/2016	525.25
Georgi Teziev Ivanov	CEO	Sales	09/12/2007	3000.00
Peter Pan Pan	Intern	Marketing	28/08/2016	599.88

Problem 19. Basic Select All Fields

Use the **softuni** database and first select all records from the **towns**, then from **departments** and finally from **employees** table. Use SQL queries and submit them to Judge at once. Submit your query statements as Prepare DB & Run queries.

Problem 20. Basic Select All Fields and Order Them

Modify queries from previous problem by sorting:

- **towns** - alphabetically by name
- **departments** - alphabetically by name
- **employees** - descending by salary

Submit your query statements as Prepare DB & Run queries.

Problem 21. Basic Select Some Fields

Modify queries from previous problem to show only **some of the columns**. For table:

- **towns** – name
- **departments** – name
- **employees** – first_name, last_name, job_title, salary

Keep the ordering from the previous problem. Submit your query statements as Prepare DB & Run queries.

Problem 22. Increase Employees Salary

Use **softuni** database and **increase the salary** of all employees by **10%.** Select **only salary** column from the **employees** table. Submit your query statements as Prepare DB & Run queries.

Problem 23. Decrease Tax Rate

Use **hotel** database and **decrease tax rate by 3%** to all payments. Select **only tax_rate** column from the **payments** table. Submit your query statements as Prepare DB & Run queries.

Problem 24. Delete All Records

Use **Hotel** database and **delete all records** from the **occupancies** table. Use SQL query. Submit your query statements as Run skeleton, run queries & check DB.