# Exercises: Table Relations

This document defines the **exercise assignments** for the ["Databases Basics - MySQL" course @ Software University.](https://softuni.bg/trainings/1443/databases-basics-mysql-september-2016)

## One-To-One Relationship

Create two tables as follows. Use appropriate data types.

|  |  |  |  |
| --- | --- | --- | --- |
| **persons** | | | |
| **person\_id** | **first\_name** | **salary** | **passport\_id** |
| 1 | Roberto | 43300.00 | 102 |
| 2 | Tom | 56100.00 | 103 |
| 3 | Yana | 60200.00 | 101 |

|  |  |
| --- | --- |
| **passports** | |
| **passport** | **passport\_number** |
| 101 | N34FG21B |
| 102 | K65LO4R7 |
| 103 | ZE657QP2 |

Insert the data from the example above.

Alter table persons and make person\_ida primary key. Create a foreign key between persons and passports by using passport\_id column.

Submit your queries by using **MySQL run skeleton run queries and check db.**

## One-To-Many Relationship

Create two tables as follows. Use appropriate data types.

|  |  |  |
| --- | --- | --- |
| **manufacturers** | | |
| **manufacturer\_id** | **name** | **established\_on** |
| 1 | BMW | 07/03/1916 |
| 2 | Tesla | 01/01/2003 |
| 3 | Lada | 01/05/1966 |

|  |  |  |
| --- | --- | --- |
| **models** | | |
| **model\_id** | **name** | **manufacturer\_id** |
| 101 | X1 | 1 |
| 102 | i6 | 1 |
| 103 | Model S | 2 |
| 104 | Model X | 2 |
| 105 | Model 3 | 2 |
| 106 | Nova | 3 |

Insert the data from the example above. Add primary keys and foreign keys.

Submit your queries by using **MySQL run skeleton run queries and check db.**

## Many-To-Many Relationship

Create three tables as follows. Use appropriate data types.

|  |  |
| --- | --- |
| **students** | |
| **student\_id** | **name** |
| 1 | Mila |
| 2 | Toni |
| 3 | Ron |

|  |  |
| --- | --- |
| **exams** | |
| **exam\_id** | **name** |
| 101 | Spring MVC |
| 102 | Neo4j |
| 103 | Oracle 11g |

|  |  |
| --- | --- |
| **students\_exams** | |
| **student\_id** | **exam\_id** |
| 1 | 101 |
| 1 | 102 |
| 2 | 101 |
| 3 | 103 |
| 2 | 102 |
| 2 | 103 |

Insert the data from the example above.  
Add primary keys and foreign keys. Have in mind that table StudentsExams should have a composite primary key.

Submit your queries by using **MySQL run skeleton run queries and check db.**

## Self-Referencing

Create a single table as follows. Use appropriate data types.

|  |  |  |
| --- | --- | --- |
| **teachers** | | |
| **teacher\_id** | **name** | **manager\_id** |
| 101 | John |  |
| 102 | Maya | 106 |
| 103 | Silvia | 106 |
| 104 | Ted | 105 |
| 105 | Mark | 101 |
| 106 | Greta | 101 |

Insert the data from the example above. Add primary keys and foreign keys. The foreign key should be between ManagerId and TeacherId.

Submit your queries by using **MySQL run skeleton run queries and check db.**

## Online Store Database

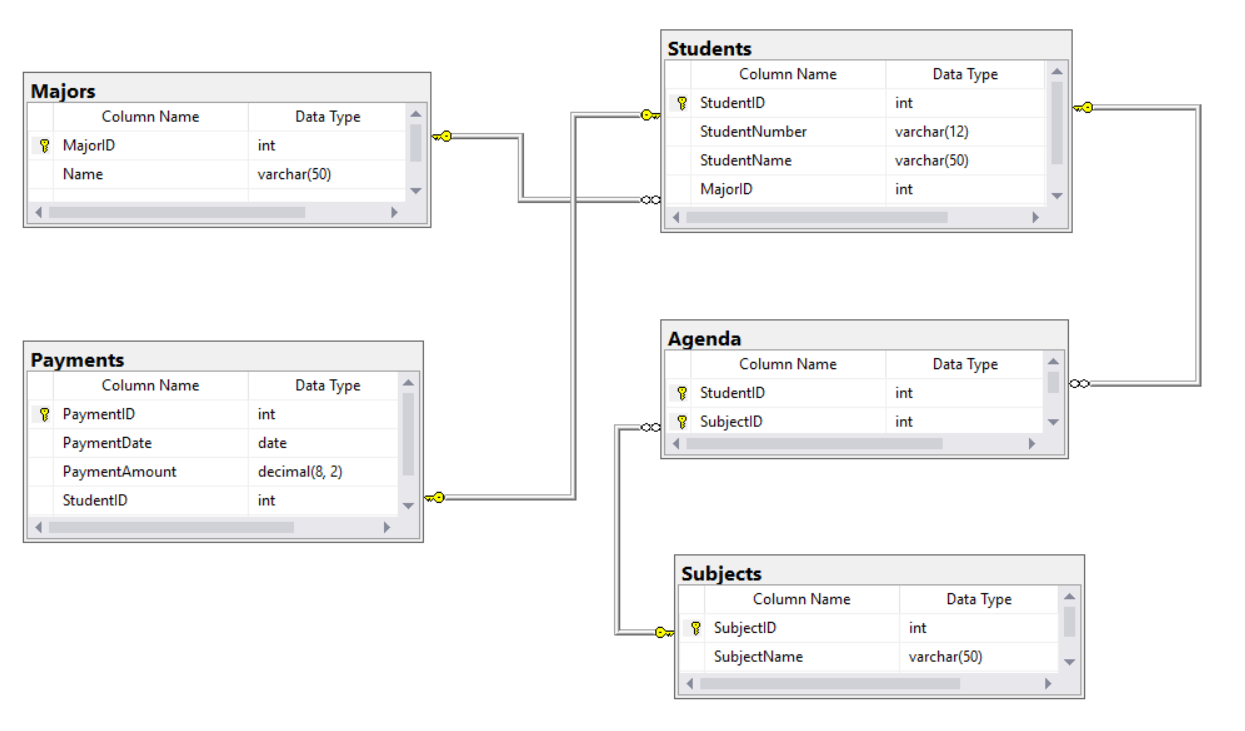
Create a new database and design the following structure:



Submit your queries by using **MySQL run skeleton run queries and check db.**

## University Database

Create a new database and design the following structure:



Submit your queries by using **MySQL run skeleton run queries and check db.**

## Soft\_Uni Design

Create an E/R Diagram of the SoftUni Database.

## Geography Design

Create an E/R Diagram of the Geography Database.

## Employee Address

Write a query that selects:

* employee\_id
* job\_title
* address\_id
* address\_text

Return the first 5 rows sorted by address\_id in ascending order.

### Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **employee\_id** | **job\_title** | **address\_id** | **address\_text** |
| 142 | Production Technician | 1 | 108 Lakeside Court |
| … | … | … | … |

## Employee Departments

Write a query that selects:

* employee\_id
* first\_name
* salary
* department\_name

Filter only employees with salary higher than 15000. Return the first 5 rows sorted by DepartmentID in ascending order.

### Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **employee\_id** | **first\_name** | **salary** | **department\_name** |
| 3 | Roberto | 43300.00 | Engineering |
| … | … | … | … |

## Employees Without Project

Write a query that selects:

* employee\_id
* first\_name

Filter only employees without a project. Return the first 3 rows sorted by employee\_id in ascending order.

### Example:

|  |  |
| --- | --- |
| **employee\_id** | **first\_name** |
| 2 | Kevin |
| … | … |

## Employees with Project

Write a query that selects:

* employee\_Id
* first\_name
* project\_name

Filter only employees with a project which has started after 13.08.2002 and it is still ongoing (no end date). Return the first 5 rows sorted by employee\_id in ascending order.

### Example

|  |  |  |
| --- | --- | --- |
| **employee\_id** | **first\_name** | **project\_name** |
|  |  |  |
| … | … | … |

## Employee 24

Write a query that selects:

* employee\_id
* first\_name
* project\_name

Filter all the projects of employee with id 24. If the project has started after 2005 the return value should be NULL.

### Example

|  |  |  |
| --- | --- | --- |
| **employee\_id** | **first\_name** | **project\_name** |
| 24 | David | NULL |
| 24 | David | Road-650 |
| … | … | … |

## Employee Manager

Write a query that selects:

* employee\_id
* first\_name
* manager\_id
* manager\_name

Filter all employees with a manager who has ID equals to 3 or 7. Return the all rows sorted by employee\_id in ascending order.

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **employee\_id** | **first\_name** | **manager\_id** | **manager\_name** |
| 4 | Rob | 3 | Roberto |
| … | … | … | … |

## Highest Peak in Bulgaria

Write a query that selects:

* country\_code
* mountan\_range
* peak\_name
* elevation

Filter all peaks in Bulgaria with elevation over 2835. Return the all rows sorted by elevation in descending order.

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **country\_code** | **mountan\_range** | **peak\_name** | **elevation** |
| BG | Rila | Musala | 2925 |
| … | … | … | … |

## Count Mountain Ranges

Write a query that selects:

* country\_code
* mountan\_ranges

Filter the count of the mountain ranges in the USA, Russia and Bulgaria.

### Example

|  |  |
| --- | --- |
| **country\_code** | **mountan\_ranges** |
| BG | 6 |
| … | … |

## Countries with Rivers

Write a query that selects:

* country\_name
* river\_name

Find the first 5 countries with or without rivers in Africa. Sort them by country\_name in ascending order.

### Example

|  |  |
| --- | --- |
| **country\_name** | **river\_name** |
| Algeria | Niger |
| Angola | Congo |
| Benin | Niger |
| Botswana | NULL |
| Burkina Faso | Niger |

## \*Continents and Currencies

Write a query that selects:

* continent\_code
* currency\_code
* currency\_usage

Find all continents and their most used currency. Filter any currency that is used in only one country. Sort your results by continent\_code.

### Example

|  |  |  |
| --- | --- | --- |
| **continent\_code** | **currency\_code** | **currency\_usage** |
| AF | XOF | 8 |
| AS | AUD | 2 |
| AS | ILS | 2 |
| EU | EUR | 26 |
| NA | XCD | 8 |
| OC | USD | 8 |

## Countries Without any Mountains

Write a query that selects **country\_code.** Find all the count of all countries which don’t have a mountain.

### Example

|  |
| --- |
| **country\_code** |
| 231 |

## Highest Peak and Longest River by Country

For each country, find the elevation of **the highest peak** and **the length of the longest river**, sorted by the highest peak elevation (from highest to lowest), then by the longest river length (from longest to smallest), then by country name (alphabetically). Display **NULL** when no data is available in some of the columns. Submit for evaluation the result grid with headers. Limit only the first 5 rows.

|  |  |  |
| --- | --- | --- |
| **country\_name** | **highest\_peak\_elevation** | **longest\_river\_length** |
| China | 8848 | 6300 |
| India | 8848 | 3180 |
| Nepal | 8848 | 2948 |
| Pakistan | 8611 | 3180 |
| Argentina | 6962 | 4880 |
| Chile | 6962 | NULL |
| … | … | … |

## \*\* Highest Peak Name and Elevation by Country

For each country, find the name and elevation of **the highest peak**, along with its mountain. When no peaks are available in some country, display elevation **0**, "**(no highest peak)**" as peak name and "**(no mountain)**" as mountain name. When multiple peaks in some country have the same elevation, display all of them. Sort the results by country name alphabetically, then by highest peak name alphabetically. Submit for evaluation the result grid with headers. Limit only the first 5 rows.

|  |  |  |  |
| --- | --- | --- | --- |
| **country** | **highest\_peak\_name** | **highest\_peak\_elevation** | **mountain** |
| Afghanistan | (no highest peak) | 0 | (no mountain) |
| … | … | … | … |
| Argentina | Aconcagua | 6962 | Andes |
| … | … | … | … |
| Bulgaria | Musala | 2925 | Rila |
| Burkina Faso | (no highest peak) | 0 | (no mountain) |
| … | … | … | … |
| United States | Mount McKinley | 6194 | Alaska Range |
| … | … | … | … |
| Zambia | (no highest peak) | 0 | (no mountain) |
| Zimbabwe | (no highest peak) | 0 | (no mountain) |