# Exercises: Built-in Functions

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

# Part I – Queries for SoftUni Database

## Find Names of All Employees by First Name

Write a SQL query to find **first** and **last names** of all employees whose **first name starts with** **“Sa” (case insensitively). Order the information by id.** Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| Sariya | Harnpadoungsataya |
| Sandra | Reategui Alayo |
| … | … |

## Find Names of All employees by Last Name

Write a SQL query to find **first** and **last names** of all employees whose **last name contains “ei” (case insensitively). Order the information by id**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| Kendall | Keil |
| Christian | Kleinerman |
| … | … |

## Find First Names of All Employees

Write a SQL query to find the **first names** of all employees in the **departments** with **ID 3 or 10** and whose **hire year** is **between 1995 and 2005 inclusively**. **Order the information by id**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |
| --- |
| **first\_name** |
| Deborah |
| Wendey |
| Candy |
| … |

## Find All Employees Except Engineers

Write a SQL query to find the **first** and **last names** of all employees whose **job titles does not contain** “**engineer**”. **Order the information by id**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| Guy | Gilbert |
| Kevin | Brown |
| Rob | Walters |
| … | … |

## Find Towns with Name Length

Write a SQL query to find **town** **names** that are **5** or **6 symbols long** and **order** them **alphabetically by town name**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |
| --- |
| **name** |
| Berlin |
| Duluth |
| Duvall |
| … |

## Find Towns Starting With

Write a SQL query to find all towns that **start with** letters **M**, **K**, **B** or **E (case insensitively).** Order them **alphabetically** by **town** **name**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **town\_id** | **name** |
| 5 | Bellevue |
| 31 | Berlin |
| 30 | Bordeaux |
| … | … |

## Find Towns Not Starting With

Write a SQL query to find all towns that **do not start with** letters **R**, **B** or **D (case insensitively).** Order them **alphabetically** by name. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **town\_id** | **name** |
| 2 | Calgary |
| 23 | Cambridge |
| 15 | Carnation |
| … | … |

## Create View Employees Hired After 2000 Year

Write a SQL query to create view **v\_employees\_hired\_after\_2000** with **the** **first and the last name** of all employees **hired after 2000 year.** Submit your query statements as **Run skeleton, run queries & check DB.**

### Example

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| Steven | Selikoff |
| Peter | Krebs |
| Stuart | Munson |
| ... | ... |

## Length of Last Name

Write a SQL query to find the names of all employees whose last name is **exactly** **5 characters long.**

### Example

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| Kevin | Brown |
| Terri | Duffy |
| Jo | Brown |
| Diane | Glimp |
| … | … |

# Part II – Queries for Geography Database

## Countries Holding ‘A’ 3 or More Times

Find all countries that hold the **letter 'A'** in their name **at least 3 times (case insensitively), sorted by ISO code**. **Display** the **country name** and the **ISO code**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **country\_name** | **iso\_code** |
| Afghanistan | AFG |
| Albania | ALB |
| … | … |

## Mix of Peak and River Names

**Combine all peak names with all river names**, so that the last letter of each peak name is the same as the first letter of its corresponding river name**. Display** the **peak name**, the **river name**, and the **obtained mix**. **Sort the results by the obtained mix alphabetically**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |  |
| --- | --- | --- |
| **peak\_name** | **river\_name** | **mix** |
| Aconcagua | Amazon | aconcaguamazon |
| Aconcagua | Amur | aconcaguamur |
| Banski Suhodol | Lena | banski suhodolena |
| … | … | … |

# Part III – Queries for Diablo Database

## Games from 2011 and 2012 year

Find the **top 50 games** **ordered by start date**, then **by name**. Display only the **games from the years 2011 and 2012**. Display the start date in the format “**YYYY-MM-DD**”. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **name** | **start** |
| Rose Royalty | 2011-01-05 |
| London | 2011-01-13 |
| Broadway | 2011-01-16 |
| … | … |

## User Email Providers

Find information about the email providers of all users. Display the **user\_name** and the **email provider**. Sort the results by **email provider alphabetically**, then by **username**. Submit your query statements **as Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **user\_name** | **Email Provider** |
| Pesho | abv.bg |
| monoxidecos | astonrasuna.com |
| bashsassafras | balibless |
| … | … |

## Get Users with IP Address Like Pattern

Find the **user\_name** and the **ip\_address** for each user, sorted by **user\_name alphabetically**. Display only the rows, where the **ip\_address** matches the pattern: “**\_\_\_.1%.%.\_\_\_**”. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |
| --- | --- |
| **user\_name** | **ip\_address** |
| bindbawdy | 192.157.20.222 |
| evolvingimportant | 223.175.227.173 |
| inguinalself | 255.111.250.207 |
| … | … |

## Show All Games with Duration and Part of the Day

Find all **games** with their corresponding **part of the day** and **duration**. **Parts of the day** should be **Morning** (**start** time is >= 0 and < 12), **Afternoon** (**start** time is >= 12 and < 18), **Evening** (**start** time is >= 18 and < 24). **Duration** should be **Extra** **Short** (smaller or equal to 3), **Short** (between 3 and 6 including), **Long** (between 6 and 10 including) and **Extra Long** in any other cases orwithout **duration**. Submit your query statements as **Prepare DB & run queries.**

### Example

|  |  |  |
| --- | --- | --- |
| **game** | **Part of the Day** | **Duration** |
| Aithusa | Evening | Short |
| Acid green | Morning | Long |
| Apple | Morning | Short |
| Broadway | Morning | Short |
| Ancalagon | Morning | Short |
| Allium drumstick | Morning | Extra Long |
| … | … | … |

# Part IV – Date Functions Queries

## Orders Table

You are given a table **orders(id, product\_name, order\_date)** filled with data. Consider that the **payment** for an order must be accomplished **within 3 days after the order date**. Also the **delivery date is up to 1 month**. Write a query to show each product’s **name**, **order date**, **pay and deliver due dates**. Submit your query statements as **Prepare DB & run queries.**

### Original Table

|  |  |  |
| --- | --- | --- |
| **id** | **product\_name** | **order\_date** |
| 1 | Butter | 2016-09-19 00:00:00 |
| 2 | Milk | 2016-09-30 00:00:00 |
| 3 | Cheese | 2016-09-04 00:00:00 |
| 4 | Bread | 2015-12-20 00:00:00 |
| 5 | Tomatoes | 2015-12-30 00:00:00 |
| … | … | … |

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **product\_name** | **order\_date** | **pay\_due** | **deliver\_due** |
| Butter | 2016-09-19 00:00:00 | 2016-09-22 00:00:00 | 2016-10-19 00:00:00 |
| Milk | 2016-09-30 00:00:00 | 2016-10-03 00:00:00 | 2016-10-30 00:00:00 |
| Cheese | 2016-09-04 00:00:00 | 2016-09-07 00:00:00 | 2016-10-04 00:00:00 |
| Bread | 2015-12-20 00:00:00 | 2015-12-23 00:00:00 | 2016-01-20 00:00:00 |
| Tomatoes | 2015-12-30 00:00:00 | 2016-01-02 00:00:00 | 2016-01-30 00:00:00 |
| … | … | … | … |