# Exercises: Built-in Functions

This document defines the **exercise assignments** for the [MySQL course @ Software University.](https://softuni.bg/opencourses/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.**

**select `first\_name`,`last\_name` from `employees` where substring(`first\_name`,1,2)='Sa'**

**order by employee\_id;**

### 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.**

**select `first\_name`,`last\_name` from `employees` where UPPER(`last\_name`) like UPPER('%EI%')**

**order by employee\_id;**

### 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.**

**select `first\_name` from `employees` where (`department\_id`=3 or `department\_id`=10) and extract(year from hire\_date) between 1995 and 2005**

**order by employee\_id;**

### 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.**

**select first\_name, last\_name from employees where `job\_title` not like '%engineer%'**

**order by employee\_id;**

### 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.**

**select name from towns where char\_length(name) between 5 and 6**

**order by name;**

### 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.**

**select town\_id, `name` from towns where upper(substring(`name`,1,1)) in ('M','K','B','E')**

**order by name;**

### 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.**

**select town\_id, `name` from towns where upper(substring(`name`,1,1)) not in ('D','R','B')**

**order by name;**

### 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.** Select all from the created view**.** Submit your queries as **Run skeleton, run queries & check DB.**

**create view v\_employees\_hired\_after\_2000 as**

**select first\_name, last\_name from employees where extract(year from hire\_date)>2000;**

**select \* from v\_employees\_hired\_after\_2000;**

### 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.**

**select first\_name, last\_name from employees where char\_length(last\_name)=5;**

### 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.**

**select country\_name,iso\_code from countries where upper(country\_name) like '%A%A%A%'**

**order by iso\_code;**

### 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(converted to lower case)**. **Sort the results by the obtained mix alphabetically**. Submit your query statements as **Prepare DB & run queries.**

**SELECT peak\_name, river\_name, CONCAT(LOWER(peak\_name), '', SUBSTRING(LOWER(river\_name), 2)) AS mix FROM peaks, rivers**

**WHERE RIGHT(peak\_name, 1) = LEFT(river\_name, 1)**

**ORDER BY mix;**

### 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.**

**select `name`,substring(`start`,1,10) as `start` from games**

**where extract(year from `start`) in (2011,2012)**

**order by `start`,`name`**

**limit 50;**

### 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.**

**select `user\_name`,substring(`email`,locate('@',`email`)+1) as `email\_provider` from users**

**order by `email\_provider`,`user\_name`**

### Example

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

## 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.**

**select `user\_name`,ip\_address from users where ip\_address like "\_\_\_.1%.%.\_\_\_"**

**order by user\_name;**

### 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.**

**select `name`,**

**case**

**when substring(`start`,12,13)>=0 and substring(`start`,12,13)<12 then 'Morning'**

**when substring(`start`,12,13)>=12 and substring(`start`,12,13)<18 then 'Afternoon'**

**when substring(`start`,12,13)>=18 and substring(`start`,12,13)<24 then 'Evening'**

**end as `Part of the Day`,**

**case**

**when duration<=3 then 'Extra Short'**

**when duration >3 and duration <=6 then 'Short'**

**when duration>6 and duration <=10 then 'Long'**

**else 'Extra Long'**

**end as `Duration` from games;**

### 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.**

**SELECT**

**`product\_name`,**

**`order\_date`,**

**DATE\_ADD(`order\_date`, INTERVAL 3 DAY) AS 'pay\_due',**

**DATE\_ADD(`order\_date`, INTERVAL 1 MONTH) AS 'deliver\_due'**

**FROM**

**`orders`;**

### 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 |
| … | … | … | … |