# Exercises: Built-in Functions

This document defines the **exercise assignments** for the ["Databases Basics - MSSQL" course @ Software University.](https://softuni.bg/trainings/2988/databases-basics-ms-sql-server-may-2020/internal#lesson-16281)

# 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**"**.**

### Example

|  |  |
| --- | --- |
| **FirstName** | **LastName** |
| 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**"**.**

### Example

|  |  |
| --- | --- |
| **FirstName** | **LastName** |
| 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 inclusive**.

### Example

|  |
| --- |
| **FirstName** |
| Deborah |
| Wendy |
| 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**".

### Example

|  |  |
| --- | --- |
| **FirstName** | **LastName** |
| 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**.

### 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**. Order them **alphabetically** by town name.

### Example

|  |  |
| --- | --- |
| **TownID** | **Name** |
| 5 | Bellevue |
| 31 | Berlin |
| 30 | Bordeaux |
| … | … |

## Find Towns Not Starting With

Write a SQL query to find all towns that **does not start with** letters **R**, **B** or **D**. Order them **alphabetically** by name.

### Example

|  |  |
| --- | --- |
| **TownID** | **Name** |
| 2 | Calgary |
| 23 | Cambridge |
| 15 | Carnation |
| … | … |

## Create View Employees Hired After 2000 Year

Write a SQL query to create view **V\_EmployeesHiredAfter2000** with **first and last name** to all employees **hired after 2000 year.**

### Example

|  |  |
| --- | --- |
| **FirstName** | **LastName** |
| 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

|  |  |
| --- | --- |
| **FirstName** | **LastName** |
| Kevin | Brown |
| Terri | Duffy |
| Jo | Brown |
| Diane | Glimp |
| … | … |

## Rank Employees by Salary

Write a query that **ranks** all employees using **DENSE\_RANK**. In the DENSE\_RANK function, employees need to be **partitioned** by **Salary** and **ordered** by **EmployeeID**. You need to find **only** the employees whose **Salary** is between 10000 and 50000 and **order** them by **Salary** in **descending** **order**.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EmployeeID** | **FirstName** | **LastName** | **Salary** | **Rank** |
| 268 | Stephen | Jiang | 48100.00 | 1 |
| 284 | Amy | Alberts | 48100.00 | 2 |
| 288 | Syed | Abbas | 48100.00 | 3 |
| … | … | … | … | … |

## Find All Employees with Rank 2 \*

Use the query from the **previous** problem and **upgrade** it, so that it finds **only** the employees whose **Rank** is 2 and again, **order** them by **Salary (descending)**.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EmployeeID** | **FirstName** | **LastName** | **Salary** | **Rank** |
| 284 | Amy | Alberts | 48100.00 | 2 |
| 292 | Martin | Kulov | 48000.00 | 2 |
| 71 | Wendy | Kahn | 43300.00 | 2 |
| … | … | … | … | … |

# Part II – Queries for Geography Database

## Countries Holding ‘A’ 3 or More Times

Find all **countries** that holds the letter 'A' in their name at least 3 times (case insensitively), **sorted by ISO code**. Display the country name and 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 names, river names, and the obtained mix (mix should be in lowercase). **Sort** the results **by** the **obtained mix**.

### Example

|  |  |  |
| --- | --- | --- |
| **PeakName** | **RiverName** | **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 of the game**. Display only **games from 2011 and 2012** year. Display start date in the format "**yyyy-MM-dd**".

### Example

|  |  |
| --- | --- |
| **Name** | **Start** |
| Rose Royalty | 2011-01-05 |
| London | 2011-01-13 |
| Broadway | 2011-01-16 |
| … | … |

## User Email Providers

**Find all users** along with information about their email providers. Display the username and email provider. **Sort the results by email provider alphabetically, then by username.**

### Example

|  |  |
| --- | --- |
| **Username** | **Email Provider** |
| Pesho | abv.bg |
| monoxidecos | astonrasuna.com |
| bashsassafras | balibless |
| … | … |

## Get Users with IPAdress Like Pattern

**Find all users** along with their IP addresses **sorted by username alphabetically**. Display only rows that IP address matches the pattern: "**\*\*\*.1^.^.\*\*\***".

Legend: **\*** - one symbol, **^** - one or more symbols  
Example

|  |  |
| --- | --- |
| **Username** | **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 part of the day and duration **sorted by game name** alphabetically **then by duration** (alphabetically, not by the timespan) and part of the day (all ascending). **Parts of the day** should be **Morning** (time is >= 0 and < 12), **Afternoon** (time is >= 12 and < 18), **Evening** (time is >= 18 and < 24). **Duration** should be **Extra** **Short** (smaller or equal to 3), **Short** (between 4 and 6 including), **Long** (greater than 6) and **Extra Long** (without duration).

### Example

|  |  |  |
| --- | --- | --- |
| **Game** | **Part of the Day** | **Duration** |
| Ablajeck | Morning | Long |
| Ablajeck | Afternoon | Short |
| Abregado Rae | Afternoon | Long |
| Abrion | Morning | Extra Short |
| Acaeria | Evening | Long |
| … | … | … |

# Part IV – Date Functions Queries

## Orders Table

You are given a table **Orders(Id, ProductName, OrderDate)** filled with data. Consider that the **payment** for that 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**.

### Original Table

|  |  |  |
| --- | --- | --- |
| **Id** | **ProductName** | **OrderDate** |
| 1 | Butter | 2016-09-19 00:00:00.000 |
| 2 | Milk | 2016-09-30 00:00:00.000 |
| 3 | Cheese | 2016-09-04 00:00:00.000 |
| 4 | Bread | 2015-12-20 00:00:00.000 |
| 5 | Tomatoes | 2015-12-30 00:00:00.000 |
| … | … | … |

### Output

|  |  |  |  |
| --- | --- | --- | --- |
| **ProductName** | **OrderDate** | **Pay Due** | **Deliver Due** |
| Butter | 2016-09-19 00:00:00.000 | 2016-09-22 00:00:00.000 | 2016-10-19 00:00:00.000 |
| Milk | 2016-09-30 00:00:00.000 | 2016-10-03 00:00:00.000 | 2016-10-30 00:00:00.000 |
| Cheese | 2016-09-04 00:00:00.000 | 2016-09-07 00:00:00.000 | 2016-10-04 00:00:00.000 |
| Bread | 2015-12-20 00:00:00.000 | 2015-12-23 00:00:00.000 | 2016-01-20 00:00:00.000 |
| Tomatoes | 2015-12-30 00:00:00.000 | 2016-01-02 00:00:00.000 | 2016-01-30 00:00:00.000 |
| … | … | … | … |

## People Table

Create a table **People(Id, Name, Birthdate).** Write a query to **find** **age in years**, **months**, **days** and **minutes** for each person for the **current time** of executing the query.

### Original Table

|  |  |  |
| --- | --- | --- |
| **Id** | **Name** | **Birthdate** |
| 1 | Victor | 2000-12-07 00:00:00.000 |
| 2 | Steven | 1992-09-10 00:00:00.000 |
| 3 | Stephen | 1910-09-19 00:00:00.000 |
| 4 | John | 2010-01-06 00:00:00.000 |
| … | … | … |

### Example Output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Age in Years** | **Age in Months** | **Age in Days** | **Age in Minutes** |
| Victor | 16 | 189 | 5754 | 8286787 |
| Steven | 24 | 288 | 8764 | 12621187 |
| Stephen | 106 | 1272 | 38706 | 55737667 |
| John | 6 | 80 | 2437 | 3510307 |
| … | … | … | … | … |