# Упражнения: Общи упражнения

## Част I – Заявки за Company\_database

## Служители със заплата над 35000

Създайте съхранена процедура **usp\_get\_employees\_salary\_above\_35000** която връща **първото и последното име на всички служители**, които имат **заплата над 35000**. Резултатът трябва да бъде сортиран по **first\_name**, а след това и по **last\_name** в азбучен ред.

### Пример

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| Amy | Alberts |
| Brian | Welcker |
| Dan | Wilson |
| … | … |

## Служители със заплата над...

Създайте съхранена процедура **usp\_get\_employees\_salary\_above**, която **приема число** като параметър и връща **първото и последното име на всички служители**, които имат заплата **равна на поне** даденото число. Резултатът трябва да се сортира по **first\_name**, а след това и по **last\_name** в азбучен ред.

### Пример

Числото в този пример е 48100.

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| Amy | Alberts |
| Brian | Welcker |
| Dylan | Miller |
| … | … |

## Градове започващи със...

Създайте съхранена процедура **usp\_get\_towns\_starting\_with** , която **приема низ като параметър** и връща **всички имена на градове започващи с този низ.** Резултатът трябва да бъде сортиран по **името на града** по азбучен ред.

### Пример

Това е списъкът с всички градове **започващи с “b”.**

|  |
| --- |
| **town\_name** |
| Bellevue |
| Berlin  |
| Bordeaux |
| Bothell |

## Служители от град

Напишете съхранена процедура **usp\_get\_employees\_from\_town** която приема **името на град** като параметър и връща **първото и последното име на всички служители, които живеят в дадения град.** Резултатът трябва да бъде сортиран по **first\_name**, а след това по **last\_name** в азбучен ред.

### Пример

Ето списък на служителите живеещи в **Sofia.**

|  |  |
| --- | --- |
| **first\_name** | **last\_name** |
| George | Denchev |
| Martin | Kulov |
| Svetlin | Nakov |

## Функция за ниво на заплата

Напишете функция **ufn\_get\_salary\_level**, която получава **заплата на служител** и връща **нивото на заплатата**.

* Ако заплатата е **< 30000** връща **“Low”**
* Ако заплатата е **между 30000 и 50000 (вкл)** връща **“Average”**
* Ако заплатата е **> 50000** връща **“High”**

### Пример

|  |  |
| --- | --- |
| **salary** | **salary\_Level** |
| 13500.00 | Low |
| 43300.00 | Average |
| 125500.00 | High |

## Дефинирайте функция

Дефинирайте функция **ufn\_is\_word\_comprised(set\_of\_letters, word)**, която връща **true** или **false** според това дали думата е съставена от даденото множество от букви.

### Пример

|  |  |  |
| --- | --- | --- |
| **set\_of\_letters** | **word** | **result** |
| Oistmiahf | Sofia | 1 |
| Oistmiahf | halves | 0 |
| Bobr | Rob | 1 |
| Pppp | Guy | 0 |

# PART II – Заявки с банковата база от данни

## Изтегляне на пари

Напишете съхранена процедура **usp\_withdraw\_money** (account\_id, money\_amount), която работи чрез транзакции.

Уверете се, че изтеглянето е приключило само ако балансът е достатъчен и **money\_amount** е валидно положително число. Работете с точност до 4-тия знак след десетичната запетая.

### Пример

Това е резултатът при **account\_id = 1** и **money\_amount = 10.**

|  |  |  |
| --- | --- | --- |
| **account\_id** | **account\_holder\_id** | **balance** |
| 1 | 1 | 113.1200 |

## Банков превод

Напишете съхранена процедура **usp\_transfer\_money(from\_account\_id, to\_account\_id, amount),** която **трансферира пари от една сметка към друга**. Разгледайте случаите, когато едно от **account\_id-тата** е невалидно, сумата от **пари е отрицателно число, балансът не е достатъчен** или трансферирате **от/към един и същ акаунт.** Уверете се, че цялата процедура **минава без грешки** и **ако се случи грешка, то това не изменя базата данни.**

### Пример

**from\_account\_id = 1, to\_account\_id = 2** и **money\_amount = 10.**

|  |  |  |
| --- | --- | --- |
| **account\_id** | **account\_holder\_id** | **balance** |
| 1 | 1 | 113.1200 |
| 2 | 3 | 4364.2300 |

## Логове за банкови сметки

Създайте нова таблица – **logs** (log\_id, account\_id, old\_sum, new\_sum). Добавете **тригер** към таблицата **accounts,** който да добавя нов запис в **logs** всеки път, когато има промяна по **банкова сметка**.

### Пример

|  |  |  |  |
| --- | --- | --- | --- |
| **log\_id** | **account\_id** | **old\_sum** | **new\_sum** |
| 1 | 1 | 123.12 | 113.12 |
| … | … | … | … |

## Тригер за мейли

Създайте нова таблица – **notification\_emails**(id, recipient, subject, body). Добавете тригер към таблицата с логовете от предната задача, за да **създадете нов мейл всеки път, когато бъде добавен запис в logs таблицата.** Следната информация е нужна за всеки мейл:

* **recipient** – account\_id
* **subject** – “Balance change for account: **{account\_id}**”
* **body** - “On **{date}** your balance was changed from **{old}** to **{new}.**”

### Пример

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **recipient** | **subject** | **body** |
| 1 | 1 | Balance change for account: 1 | On Sep 15 2016 at 11:44:06 AM your balance was changed from 133 to 143. |
| … | … | … | … |

## Министерство на образованието и науката (МОН)

* Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

* Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен** **лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).

