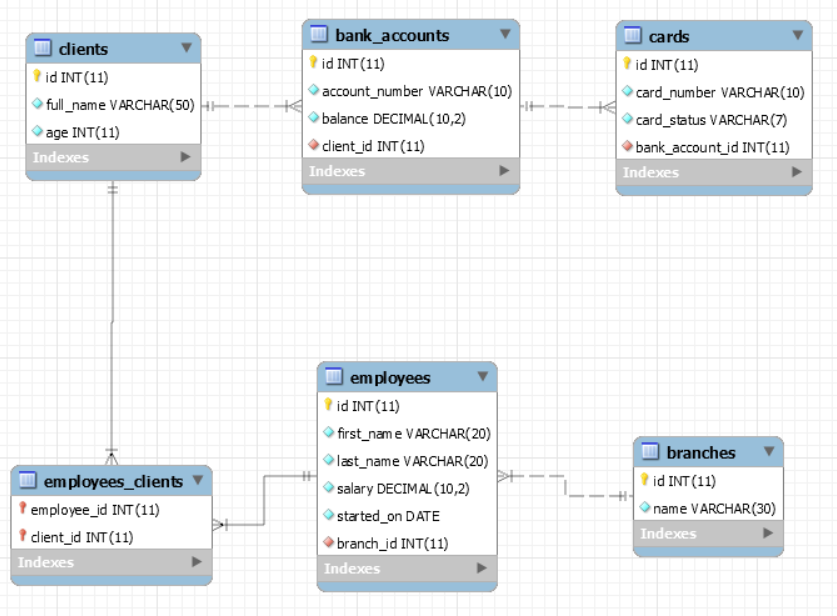
# Database Basics (MySQL) Retake Exam Royal United Kingsman – Bank

Royal United Kingsman Bank or most widely known as R.U.K. Bank is a new bank founded by Darkman Nakov. You have been employed by the bank to design a database prototype, which will lay the foundation for the main database. You will need to prove your skills in database definition, data manipulation and extraction and database programmability.

## Section 0: Database Overview

You have been given an Entity / Relationship Diagram of the Database:

****

The **Bank’s Database** needs to hold information about **branches**, **employees**, **clients**, **bank accounts**, **cards**.

Your task is to create a database called ruk\_database. Then you will have to create several **tables**.

* branches – contains information about the **branches**.
* employees – contains information about the **employees**.
  + Each employee has a branch.
* clients – contains information about the **clients**.
* employees\_clients – a **many** to **many** **mapping** table between the **employees** and the **clients**.
* bank\_accounts – contains information about the **bank accounts**.
  + Each bank\_account has a client.
* cards – contains information about the **cards**.
  + Each card has a client.
  + Each card has a bank\_account.

## Section 1: Data Definition Language (DDL) – 40 pts

Make sure you implement the whole database correctly on your local machine, so that you could work with it.

The instructions you’ll be given will be the minimal needed for you to implement the database.

### Table Design

You have been tasked to create the tables in the database by the following models:

**branches**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| name | A **string** containing a maximum of **30 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**. UNIQUE** values. |

**employees**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| first\_name | A **string** containing a maximum of **20 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| last\_name | A **string** containing a maximum of **20 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| salary | **DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |
| started\_on | A **DATE** field. Format - (**YYYY-MM-DD**). | **NULL** is **NOT** permitted**.** |
| branch\_id | **Integer,** from **1** to **2,147,483,647.** | Relationship with table branches.  **NULL** is **NOT** permitted**.** |

**client**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| full\_name | A **string** containing a maximum of **50 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| age | **Integer,** from **1** to **2,147,483,647.** | **NULL** is **NOT** permitted**.** |

**employees\_clients**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| employee\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table employees. |
| client\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table clients. |

**bank\_accounts**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| account\_number | A **string** containing a maximum of **10 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| balance | **DECIMAL**, up to **10 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |
| client\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table clients.  **NULL** is **NOT** permitted**.**  **UNIQUE** values. |

**cards**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| card\_number | A **string** containing a maximum of **19 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| card\_status | A **string** containing a maximum of **7 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| bank\_account\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table bank\_accounts.  **NULL** is **NOT** permitted**.** |

Submit your solutions in Judge on the first task. Submit **all** SQL table creation statements.

You will also be given a data.sql file. It will contain a **dataset** with random data which you will need to **store** in your **local database**. This data will be given to you so you will not have to think of data and lose essential time in the process. The data is in the form of **INSERT** statement queries.

## Section 2: Data Manipulation Language (DML) – 30 pts

Here we need to do several manipulations in the database, like changing data, adding data etc.

### Data Insertion

You will have to **INSERT** records of data into the cards table, based on the clients table.

For clients with id between **191** and **200** (**inclusive**), **insert data** in the cards table with the **following values**:

* card\_number –set it to **“**{firstName} {lastName}**”.** Where the firstName and lastName are the **split** full name of the client. Both values should be reversed!
* card\_status – set it to “Active”.
* bank\_account\_id – **Extract** the **client’s** **bank account** **id**.

### Data Update

**UPDATE** all clients which have the same id as the employee they are appointed to. Set their employee to the employee with the lowest count of clients.

If there are 2 such employees with equal count of clients, take the one with the lowest id.

### Data Deletion

R.U.K. Bank is a sophisticated network. As such, it cannot allow procrastination and lazy behavior.

**DELETE** all employees which do not have any clients.

## Section 3: Querying – 100 pts

And now we need to do some data extraction. **Note** that the **example results** from **this section** use a **fresh database**. It is **highly recommended** that you **clear** the **database** that has been **manipulated** by the **previous problems** from the **DML** **section** and **insert again** the **dataset** you’ve been given, to ensure **maximum consistency** with the **examples** given in this section.

### Clients

Extract from the database, all of the clients.

**ORDER** the results **ascending** by client id.

#### Required Columns

* id (clients)
* full\_name

#### Example

|  |  |
| --- | --- |
| **id** | **full\_name** |
| 1 | Hunter Wesgate |
| ... | ... |

### Lucky Numbers

When an employee has the same id as one of his clients, it’s a lucky number.

**Extract** from the database, all of the clients, which have the **same id** as the employee that is assigned to them.

**ORDER** the results **ascending** by client id.

#### Required Columns

* employee\_id
* client\_id

#### Example

|  |  |
| --- | --- |
| **employee\_id** | **client\_id** |
| 27 | 27 |
| ... | ... |

### Newbies

One of your bosses has requested a functionality which checks the newly employed – highly paid people.

Extract from the database, all of the employees, which have salary **greater than** or **equal** to 100000 and have started **later than** or **equal** to **the 1st of January – 2018**.

The salary should have a “$” as a **prefix**.

**ORDER** the results **descending** by salary.

#### Required Columns

* id (employees)
* full\_name (first\_name + “ “ + last\_name)
* salary
* started\_on

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **full\_name** | **salary** | **started\_on** |
| 41 | Lisbeth Skett | $981421.79 | 2018-04-16 |
| ... | ... | ... |  |

### Cards against Humanity

Extract from the database, all of the cards, and the clients that own them, so that they end up in the following format:

{card\_number} : {full\_name}

**ORDER** the results **descending** by card id.

#### Required Columns

* id (cards)
* card\_token

#### Example

|  |  |
| --- | --- |
| **id** | **card\_token** |
| 500 | SM80 M775 4918 653X : Erin Cullingworth |
| ... | ... |

### Disabled Accounts

Extract from the database, all of the bank accounts which have all of their cards with card\_status – ‘Frozen’ or ‘Deleted’.

The salary should have a “$” as a **prefix**.

**ORDER** the results **descending** by bank account id.

#### Required Columns

* account\_number
* balance
* full\_name (clients)

#### Example

|  |  |  |
| --- | --- | --- |
| **account\_number** | **name** | **full\_name** |
| 46223600-5 | $544808.92 | Ignacio Hattersley |
| ... | ... | ... |

### Top 5 Employees

Extract from the database, the **top 5** employees, in terms of clients assigned to them.

**ORDER** the results **descending** by count of clients, and **ascending** by employee id.

#### Required Columns

* name (employees)
* started\_on
* count\_of\_clients

#### Example

|  |  |  |
| --- | --- | --- |
| **name** | **started\_on** | **count\_of\_clients** |
| Trula Glasscott | 2017-08-23 | 14 |
| ... | ... | ... |

### Family Guy

Extract from the database, the **top 1 client** in terms of count of cards which he / she has on his / her bank account.

If there are 2 clients have the **same** count of cards, **order** them **ascending**, by client id.

#### Required Columns

* id (clients)
* full\_name (clients)
* cards (count of cards)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **full\_name** | **cards** |
| 2 | Baxy David | 6 |

### Adashes

Extract from the database, every employee that has a client with the same first name as **his** / **hers**.

**ORDER** the results **ascending** by client id.

#### Required Columns

* client\_name (clients)
* employee\_name (employees)

#### Example

|  |  |
| --- | --- |
| **client\_name** | **employee\_name** |
| Norma | Norma |
| ... | ... |

### Branching Clients

Extract from the database, for every branch – the count of clients that have been assigned to its employees.

**ORDER** the results **descending** by clients (count of clients), and **ascending** by branch id.

#### Required Columns

* name (branches)
* clients (count of clients)

#### Example

|  |  |
| --- | --- |
| **id** | **clients** |
| Leroy Branch | 14 |
| ... | ... |

## Section 4: Programmability – 20 pts

The time has come for you to prove that you can be a little more dynamic on the database. So, you will have to write several procedures.

### Extract Client Info

Create a stored procedure udp\_clientinfo which accepts the following parameters:

* full\_name

And **extracts data** about the client with the **given** full name.

Aside from the full\_name, the procedure should **extract** the client’s age, bank account number and balance.

The account’s salary should have “$” **prefix**.



#### Result

|  |  |  |  |
| --- | --- | --- | --- |
| **full\_name** | **age** | **account\_number** | **balance** |
| Hunter Wesgate | 33 | 69666616-8 | $803355.32 |