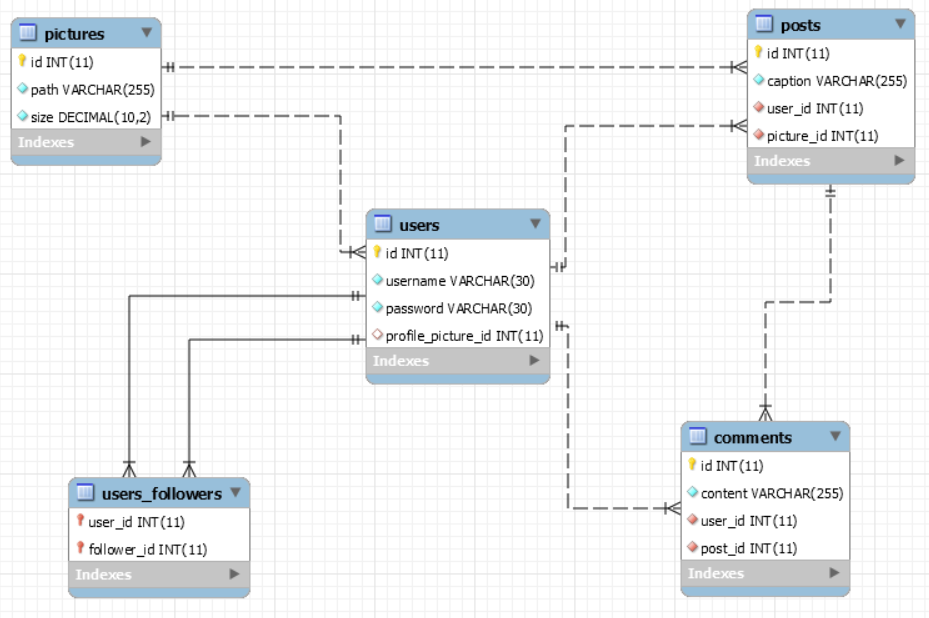
# Database Basics (MySQL) Retake Exam Instagraph

You’ve most likely heard of Instagram. Well … There is a side project called “Instagraph” which is the back-up data of Instagram. You are one of the few selected to work in the multi-billion company, as one of the back-up database managers. You’ll need to prove your skills by designing and manipulating data in the Instagraph prototype.

## Section 0: Database Overview

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

****

The Instagraph Database needs to hold information about pictures, users, posts and comments.

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

* pictures – contains information about the **pictures**.
* users – contains information about the **users**.
  + Each user may have a profile picture.
* posts – contains information about the **posts**.
  + Each post has a user.
  + Each post has a picture.
* comments – contains information about the **comments**.
  + Each comment has a user.
  + Each comment has a post.
* users\_followers – a **many** to **many** table connected to the **users**.

## 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:

**pictures**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer,** from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| path | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted**.** |
| size | **Decimal**, **up** to **10 digits**, **2** of which after the **decimal point**. | **NULL** is **NOT** permitted**.** |

**users**

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

**posts**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| caption | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| user\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users.  **NULL** is **NOT** permitted**.** |
| picture\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table pictures.  **NULL** is **NOT** permitted**.** |

**comments**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| id | **Integer**, from **1** to **2,147,483,647.** | **Primary Key AUTO\_INCREMENT** |
| content | A **string** containing a maximum of **255 characters**. Unicode is **NOT** needed. | **NULL** is **NOT** permitted. |
| user\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users.  **NULL** is **NOT** permitted**.** |
| post\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table posts.  **NULL** is **NOT** permitted**.** |

**users\_followers**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| user\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users. |
| follower\_id | **Integer**, from **1** to **2,147,483,647.** | Relationship with table users. |

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 comments table, based on the posts table. For posts with id between **1** and **10**, insert data in the comments table with the following values:

* content –set it to **“**Omg!{name}!This is so cool!**”.** Where the name is the username of the user that **posted** the post.
* user\_id – **MULTIPLY** the id of the post by **3** and **DIVIDE** it by **2**.
  + **ROUND** the resulting value **UP**.
* post\_id – the post’s id**.**

### Data Update

**UPDATE** all users which do **NOT** have a profile picture. **Set** their profile picture id to the **count** of **followers** they have. If they have 0, set it to the user’s id.

### Data Deletion

Naturally, unpopular profiles are being treated as abandoned. **DELETE** all users which do **NOT** **follow** anyone and **no one follows** them.

## 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.

### Users

Extract from the database, all of the users.

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

#### Required Columns

* id (users)
* username

#### Example

|  |  |
| --- | --- |
| **id** | **username** |
| 1 | UnderSinduxrein |
| ... | ... |

### Cheaters

Apparently, there was a bug that allowed users to follow themselves. You need to track them.

Extract from the database, all of the users, which follow themselves.

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

#### Required Columns

* id (users)
* username

#### Example

|  |  |
| --- | --- |
| **id** | **username** |
| 2 | BlaAntigadsa |
| ... | ... |

### High Quality Pictures

High quality pictures have bigger size, naturally. Extract from the database, all of the pictures, which have size, **GREATER** than **50000**, and their path contains “jpeg” or “png”.

**ORDER** the results **descending** by picture size.

#### Required Columns

* id (pictures)
* path
* size

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **path** | **size** |
| 44 | src/folders/resources/images/profile/browsed/png/841p0J24Oa.png | 73543.36 |
| ... | ... | ... |

### Comments and Users

Extract from the database, all of the comments, and the users that posted them, so that they end up in the following format:

{username} : {commentContent}

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

#### Required Columns

* id (comments)
* full\_comment

#### Example

|  |  |
| --- | --- |
| **id** | **full\_comment** |
| 50 | BlaSinduxrein : I cannot beleive this Simply amazing! Lol |
| ... | ... |

### Profile Pictures

Extract from the database, all of the users, which have the same profile picture.

Extract the size of the picture and add “KB” to the **end** of it.

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

#### Required Columns

* id (users)
* username
* size (pictures)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **username** | **size** |
| 7 | WhatTerrorBel | 44273.27KB |
| ... | ... | ... |

### Spam Posts

Extract from the database, the **top 5** posts, in terms of count of comments on them.

**ORDER** the results **descending** by comments (count of comments), and **ascending** by post id.

#### Required Columns

* id (posts)
* caption (posts)
* comments (count of comments)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **caption** | **comments** |
| 36 | #feminist #happy #ring #my #swag #gerynikol #sleepless #yolo | 4 |
| ... | ... | ... |

### Most Popular User

Extract from the database, the **most popular** user – the **1st** in terms of count of followers.

#### Required Columns

* id (users)
* username
* posts (count of posts)
* followers (count of followers)

#### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **username** | **posts** | **followers** |
| 19 | ZendArmyhow | 3 | 9 |

### Commenting Myself

Extract from the database, for every user – the count of comments he has on his posts by himself.

In other words, **extract** for each user, the count of comments he has **placed** on his own **posts**.

**ORDER** the results **descending** by my\_comments (count of comments), and **ascending** by user id.

#### Required Columns

* id (users)
* username
* my\_comments (count of comments)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **username** | **my\_comments** |
| 10 | ScoreSinduxIana | 2 |
| ... | ... | ... |

### User Top Posts

Extract from the database, for every user – the post with the **HIGHEST** count of comments on it.

If the user has **NO** posts, **IGNORE** him.

If there are 2 posts at the **top** with the **same** count of comments, **pick** the **one** with the **LOWER** id.

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

#### Required Columns

* id (users)
* username
* post (top post caption)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **username** | **post** |
| 1 | UnderSinduxrein | #gerynikol #happy #sky #epic #everything #suzanita |
| ... | ... | ... |

### Posts and Commentators

Extract from the database, for every post – the count of users that have comments on it.

**NOTE**: **1** user may have **more** than **1** comment on the post.

**ORDER** the results **descending** by users (count of users), and **ascending** by post id.

#### Required Columns

* id (posts)
* caption
* users (count of users)

#### Example

|  |  |  |
| --- | --- | --- |
| **id** | **caption** | **users** |
| 36 | #feminist #happy #ring #my #swag #gerynikol #sleepless #yolo | 4 |
| ... | ... | ... |

## Section 4: Programmability – 30 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.

### Post

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

* username
* password
* caption
* path

And checks the following things:

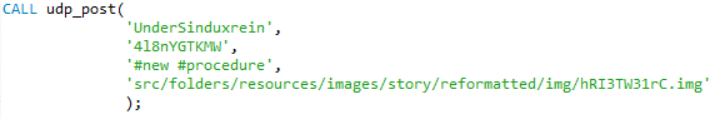
If the password does **NOT** match the username in the users table:

Throw an exception with error code ‘45000’ and message ‘Password is incorrect!’.

If there is no picture with the given path in the pictures table:

Throw an exception with error code ‘45000’ and message ‘The picture does not exist!’.

If **all checks pass**, extract the id of the corresponding user, from the users table, then the picture id from the pictures table and **INSERT** a new post into the posts table with the extracted data.



#### Result

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **caption** | **user\_id** | **picture\_id** |
| ... | ... |  | ... |
| 41 | #new #procedure | 1 | 45 |

### Filter

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

* hashtag

And extracts all posts that **CONTAIN** the **given** **hashtag** in their caption.

The procedure should **extract** the user’s username.

The **hashtag** will be given **WITHOUT** the ‘#’ sign.

The posts should be ordered **ascending** by post id.



#### Result

|  |  |  |
| --- | --- | --- |
| **id** | **caption** | **user** |
| 2 | #cool #justdoit #sky #ocean #reason #feminist #gram #faith #hope #insta | HighAsmahow |
| 7 | #cool #suzanita #the #dawn #my | HighAsmahow |
| ... | ... |  |