Advanced Database Feature Queries

Sally Xue, Pradyumann Singhal, Lily Zhang, Joseph Schanne

1. Stored Procedures

a. Update to a new password

```
DELIMITER //
CREATE PROCEDURE UpdatePassword(
IN tmpUserID varchar(255),
IN oldPassword varchar(255),
IN newPassword varchar(255))
BEGIN
        DECLARE Counter INT;
        DECLARE old_password_db VARCHAR(40);
        DECLARE storedID VARCHAR(40);
SELECT COUNT(*) as cnt INTO Counter
      FROM USERS u
      WHERE u.UserID = tmpUserID
      LIMIT 1;
SELECT UserID as usersId, Password INTO storedID, old_password_db
      FROM USERS u
      WHERE u.UserID = tmpUserID
      LIMIT 1;
IF Counter = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No user entry with
specified user id found!!';
ELSEIF oldPassword  = old_password_db THEN
        UPDATE USERS u
        SET Password = NewPassword
        WHERE u.UserID = tmpUserID;
ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Old password is
        wrong!!';
END IF;
END;
//
DELIMITER ;
```

b. Add Comment Under a song

```
DELIMITER //
```

```
CREATE PROCEDURE AddCommentProcedure(
IN tmpUserID varchar(255),
IN tmpSongID varchar(255),
IN tmpCommentInfo varchar(255),
IN tmpRating int,
IN tmpResponseTo varchar(255)
)
BEGIN
DECLARE LastCommentTime VARCHAR(255);
SELECT CreatedOn INTO LastCommentTime
FROM COMMENTS
WHERE SongID = tmpSongID  AND UserID = tmpUserID
ORDER BY CreatedOn DESC
LIMIT 1;

IF LastCommentTime IS NOT NULL AND TIMESTAMPDIFF(MINUTE,
lastCommentTime, NOW()) < 5 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'You can only comment once every 5 minutes on the
same song!'; END IF;
END;
//
DELIMITER ;
```

2. **Triggers**
   a. **Insert a Dangerous Password**

```
DELIMITER //

CREATE TRIGGER insertDangerousPassword BEFORE INSERT ON USERS
FOR EACH ROW
BEGIN
   IF NEW.Password LIKE '%"%' OR NEW.Password LIKE "%'%" THEN
         SIGNAL SQLSTATE '45000'
         SET MESSAGE_TEXT = 'Dangerous characters are being inserted in the
         password!!';
    END IF;
END;
//
DELIMITER ;
```

   b. **Update to a Dangerous Password**

```
DELIMITER //
CREATE TRIGGER updateToDangerousPassword BEFORE UPDATE ON
USERS
FOR EACH ROW
BEGIN
   IF NEW.Password LIKE '%"%' OR NEW.Password LIKE "%'%" THEN
         SIGNAL SQLSTATE '45000'
         SET MESSAGE_TEXT = 'Dangerous characters are being updated in the
         password!!';
    END IF;
END;
//
DELIMITER ;
```

3. Constraints

a. Password Length Requirement

```
ALTER TABLE USERS ADD CONSTRAINT MinLengthPasswordCheck
CHECK(LENGTH(password) >= 5);
```

4. Transactions

a. Edit Comment Time Limit Check

```
     cursor.execute("START TRANSACTION;")

     # Check last edit timestamp
     query_check = "SELECT LastEditTime FROM COMMENTS WHERE
CommentID = %s;"
     cursor.execute(query_check, (comment_id,))
     result = cursor.fetchone()
     if result and result[0]:
        last_edit_timestamp = result[0].timestamp()
        current_timestamp = datetime.now().timestamp()
        if current_timestamp - last_edit_timestamp < 5:
           cursor.execute("ROLLBACK;")
           return jsonify({"error": "Rate limit exceeded. Please wait 5 seconds
before editing again."}), 429

     # Proceed with editing
     current_time = datetime.now()
     query_update = """
     UPDATE COMMENTS SET CommentInfo = %s, Rating = %s, LastEditTime
= %s
     WHERE CommentID = %s;
```

```
    """
    cursor.execute(query_update, (new_comment_info, new_rating,
current_time, comment_id))
    connection.commit()
    return jsonify(True)

  except Exception as e:
    connection.rollback()
    return jsonify({"error": str(e)}), 500
```

b. Create a New User with unique userID

```
cursor = connection.cursor()
cursor.execute("START TRANSACTION;")
    # Check if username already exists
    cursor.execute("SELECT Username FROM USERS WHERE Username =
%s", (username,))
    if cursor.fetchone():
      cursor.execute("ROLLBACK;")
      cursor.close()
      return jsonify({"error": "Username already exists"}), 409

    # Check if email already exists
    cursor.execute("SELECT Email FROM USERS WHERE Email = %s",
(email,))
    if cursor.fetchone():
      cursor.execute("ROLLBACK;")
      cursor.close()
      return jsonify({"error": "Email already exists"}), 409

    # Generate a unique UserID
    cursor.execute("SELECT MAX(CAST(UserID AS SIGNED)) FROM USERS
WHERE UserID REGEXP '^[0-9]+$'")
    result = cursor.fetchone()
    next_id = str(1 if result[0] is None else result[0] + 1)

    print(f"Generated UserID: {next_id}")  # Debug print

    # Insert new user
    query = "INSERT INTO USERS (UserID, Username, Password, Email)
VALUES (%s, %s, %s, %s)"
    print(f"Executing query with values: ({next_id}, {username}, {password},
{email})")  # Debug print
    cursor.execute(query, (next_id, username, password, email))
```

```
connection.commit()
cursor.close()
```