Stored Procedure Documentation

1. ChangeUserPassword.sql

```sql
CREATE PROCEDURE ChangeUserPassword
    @UserID INT,
    @NewPasswordHash VARCHAR(50),
    @Message VARCHAR(100) OUTPUT
AS
BEGIN
    IF LEN(@NewPasswordHash) = 0
    BEGIN
        SET @Message = 'Password cannot be empty.';
        RETURN;
    END

    UPDATE Users
    SET PasswordHash = @NewPasswordHash
    WHERE UserID = @UserID;

    SET @Message = 'Password updated successfully.';
END;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-05-29T20:02:43.6444267+05:00
```

Purpose:
This procedure is designed to **safely update the password** for a user in the `Users` table. It takes the user's ID and the new password as input and updates the record.

Description:
The procedure is helpful in allowing users to change their passwords when needed. Once executed, it confirms the action using a `PRINT` statement, which could be useful for front-end acknowledgment.

Output when tested:

```sql
DECLARE @msg VARCHAR(100);
EXEC ChangeUserPassword @UserID = 1, @NewPasswordHash = 'newhashedpassword123', @Message = @msg OUTPUT;
SELECT @msg AS ResultMessage;
```

100 %

Results | Messages

	ResultMessage
1	Password updated successfully.

2.DeleteCategory.sql

backupQuery.sql -...P9G70\E N V Y (54))* | SQLQuery24.sql - D...9G70\E N V Y (5

```sql
CREATE PROCEDURE DeleteCategory
    @CategoryID INT
AS
BEGIN
    DELETE FROM Categories
    WHERE CategoryID = @CategoryID;

    PRINT 'Category deleted successfully.';
END;


EXEC DeleteCategory @CategoryID = 3;
```

100 %

Messages

```
(0 rows affected)
Category deleted successfully.

Completion time: 2025-05-28T13:45:39.8761396+05:00
```

2.

Purpose:
This procedure **deletes a category** from the `Categories` table based on the provided CategoryID.

Description:
This is typically used in an admin panel where categories need to be managed. It ensures the deletion of unwanted or deprecated categories. It prints a message on successful deletion.

Output when tested:

```
EXEC DeleteCategory @CategoryID = 3;
```

100 %

Messages

```
(0 rows affected)
Category deleted successfully.

Completion time: 2025-05-29T20:05:57.6564939+05:00
```

3.DeleteSkill.sql

```
CREATE PROCEDURE DeleteSkill
    @SkillID INT
AS
BEGIN
    DELETE FROM Skills
    WHERE SkillID = @SkillID;
END;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-05-29T20:08:47.7083883+05:00
```

Purpose:
This procedure **removes a skill** from the `Skills` table by specifying the SkillID.

Description:
Used when an obsolete or incorrect skill needs to be removed. It's simple, to-the-point, and provides a confirmation message.

Output when tested:

```sql
EXEC DeleteSkill @SkillID = 1;
```

100 %

Messages

```
(1 row affected)

Completion time: 2025-05-29T20:17:44.7156641+05:00
```

4.GetUserByID.sql

```sql
CREATE PROCEDURE GetUserByID
    @UserID INT
AS
BEGIN
    SELECT UserID, Username, Email, UserType, RegistrationDate, LastLoginDate, ProfilePictureURL, Bio
    FROM Users
    WHERE UserID = @UserID;
END;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-05-29T20:19:37.0964473+05:00
```

Purpose:
This procedure **retrieves user details** from the `Users` table based on the given UserID.

Description:
Useful for front-end or admin views where user profiles or account details are to be shown. Instead of returning a message, it returns a row of data from the Users table.

Output when tested:

```sql
EXEC GetUserByID @UserID = 1;
```

100 %

Results | Messages

	UserID	Username	Email	UserType	RegistrationDate	LastLoginDate	ProfilePictureURL	Bio
1	1	ahsan_ali	ahsan.ali@example.com	Freelancer	2024-01-15	2025-05-15	ahsan.jpg	Creative web developer from Lahore.

5. UpdateCategoryName.sql

```sql
CREATE PROCEDURE UpdateCategoryName
    @CategoryID INT,
    @CategoryName VARCHAR(50)
AS
BEGIN
    UPDATE Categories
    SET CategoryName = @CategoryName
    WHERE CategoryID = @CategoryID;

    PRINT 'Category name updated successfully.';
END;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-05-29T20:21:38.5695844+05:00
```

Purpose:
This procedure **updates the name** of a category using the CategoryID.

Description:
When a category name needs correction or rebranding, this procedure allows quick updates. It is minimal and useful for frontend control panels.

Output when tested:

```sql
EXEC UpdateCategoryName @CategoryID = 2, @CategoryName = 'Web Development';
```

100 %

Messages

```
(0 rows affected)
Category name updated successfully.

Completion time: 2025-05-29T20:23:40.5535017+05:00
```

6. UpdateLastLogin.sql

```sql
CREATE PROCEDURE UpdateLastLogin
    @UserID INT,
    @Message VARCHAR(100) OUTPUT
AS
BEGIN
    UPDATE Users
    SET LastLoginDate = GETDATE()
    WHERE UserID = @UserID;

    SET @Message = 'Last login date updated successfully.';
END;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-05-29T20:24:15.3359669+05:00
```

Purpose:
This procedure **updates the LastLogin column** in the `Users` table for a specific user.

Description:
It can be triggered after user login to update the login timestamp, helpful for user activity tracking. The procedure uses `GETDATE()` to insert the current date and time.

Output when tested:

```sql
DECLARE @msg VARCHAR(100);
EXEC UpdateLastLogin @UserID = 1, @Message = @msg OUTPUT;
SELECT @msg AS ResultMessage;
```

100 %

Results | Messages

	ResultMessage
1	Last login date updated successfully.

---

7. UpdateSkillDescription.sql

```sql
CREATE PROCEDURE UpdateSkillDescription
    @SkillID INT,
    @Description VARCHAR(50)
AS
BEGIN
    UPDATE Skills
    SET Description = @Description
    WHERE SkillID = @SkillID;
END;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-05-29T20:26:22.0705236+05:00
```

Purpose:
This procedure **updates the description** of a specific skill in the `Skills` table.

Description:
It helps in maintaining and refining the description text for skills shown to clients or freelancers. The update is followed by a confirmation message.

Output when tested:

```sql
EXEC UpdateSkillDescription @SkillID = 1, @Description = 'Updated description';
```

100 %

Messages

```
(0 rows affected)

Completion time: 2025-05-29T20:28:11.2628394+05:00
```