USE master;

GO

-- Koppla från och ta bort databasen om den existerar

IF EXISTS (SELECT \* FROM sys.databases WHERE name = 'HederligeHarrysBilar')

BEGIN

ALTER DATABASE HederligeHarrysBilar

SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

DROP DATABASE HederligeHarrysBilar;

PRINT 'Databasen "HederligeHarrysBilar" har tagits bort.';

END;

-- Skapa databasen

CREATE DATABASE HederligeHarrysBilar;

PRINT 'Databasen "HederligeHarrysBilar" har skapats.';

GO

-- Växla till databasen

USE HederligeHarrysBilar;

GO

-- Ta bort tabellerna om de redan existerar

IF OBJECT\_ID('Users', 'U') IS NOT NULL

BEGIN

DROP TABLE Users;

END;

GO

IF OBJECT\_ID('LoginAttempts', 'U') IS NOT NULL

BEGIN

DROP TABLE LoginAttempts;

END;

GO

-- Skapa tabellen Users

CREATE TABLE Users (

UserID INT IDENTITY(1,1) PRIMARY KEY,

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

EmailAddress NVARCHAR(255) NOT NULL UNIQUE,

PasswordHash NVARCHAR(128) NOT NULL,

Salt NVARCHAR(50) NOT NULL,

Address NVARCHAR(100) NOT NULL,

PostalCode NVARCHAR(20) NOT NULL,

City NVARCHAR(50) NOT NULL,

Country NVARCHAR(50) NOT NULL,

PhoneNumber NVARCHAR(20) NOT NULL,

IsVerified BIT NOT NULL DEFAULT 0,

IsLockedOut BIT NOT NULL DEFAULT 0,

Role NVARCHAR(20) NOT NULL DEFAULT 'Customer',

PasswordResetCode NVARCHAR(255) NULL,

PasswordResetExpiry DATETIME NULL,

CreatedAt DATETIME NOT NULL DEFAULT GETDATE(),

UpdatedAt DATETIME NOT NULL DEFAULT GETDATE()

);

GO

-- Skapa tabellen LoginAttempts

CREATE TABLE LoginAttempts (

AttemptID INT IDENTITY(1,1) PRIMARY KEY,

UserID INT NULL,

Email NVARCHAR(255) NULL,

IPAddress NVARCHAR(50) NOT NULL,

AttemptDate DATETIME NOT NULL DEFAULT GETDATE(),

Success BIT NOT NULL,

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

-- Index för snabbare sökning på e-postadress

CREATE INDEX IDX\_Email ON Users (EmailAddress);

-- Index för snabbare sökning på IP-adress i LoginAttempts

CREATE INDEX IDX\_IP ON LoginAttempts (IPAddress);

GO

--Registrering av användare

CREATE OR ALTER PROCEDURE RegisterUser

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@EmailAddress NVARCHAR(255),

@Password NVARCHAR(50),

@Address NVARCHAR(255),

@PostalCode NVARCHAR(20),

@City NVARCHAR(50),

@Country NVARCHAR(50),

@PhoneNumber NVARCHAR(20)

AS

BEGIN

-- Kontrollera om e-postadressen redan finns

IF EXISTS (SELECT 1 FROM Users WHERE EmailAddress = @EmailAddress)

BEGIN

PRINT 'E-postadressen används redan.';

RETURN;

END

-- Kontrollera lösenordets styrka

IF @Password NOT LIKE '%[A-Z]%' OR -- Minst en stor bokstav

@Password NOT LIKE '%[a-z]%' OR -- Minst en liten bokstav

@Password NOT LIKE '%[0-9]%' OR -- Minst en siffra

@Password NOT LIKE '%[^a-zA-Z0-9]%' OR -- Minst ett specialtecken

LEN(@Password) < 8 -- Minsta längd

BEGIN

PRINT 'Lösenordet uppfyller inte kraven.';

RETURN;

END

-- 🔹 Generera ett slumpmässigt salt

DECLARE @Salt NVARCHAR(255) = CONVERT(NVARCHAR(255), NEWID());

-- 🔹 Kombinera lösenord + salt och hash med SHA2\_256

DECLARE @PasswordWithSalt NVARCHAR(255) = @Password + @Salt;

DECLARE @PasswordHash NVARCHAR(128) = CONVERT(NVARCHAR(255), HASHBYTES('SHA2\_256', @PasswordWithSalt), 1);

-- 🔹 Lägg till användaren i databasen

INSERT INTO Users (FirstName, LastName, EmailAddress, Salt, PasswordHash, Address, PostalCode, City, Country, PhoneNumber, Role)

VALUES (@FirstName, @LastName, @EmailAddress, @Salt, @PasswordHash, @Address, @PostalCode, @City, @Country, @PhoneNumber, 'Customer');

PRINT 'Användaren har registrerats!';

END;

GO

--Uppdatering av roll (endast av Admins)

CREATE OR ALTER PROCEDURE UpdateUserRole

@EmailAddress NVARCHAR(255),

@NewRole NVARCHAR(20)

AS

BEGIN

-- Kontrollera om användaren finns

IF NOT EXISTS (SELECT 1 FROM Users WHERE EmailAddress = @EmailAddress)

BEGIN

PRINT 'Användaren hittades inte.';

RETURN;

END

-- Kontrollera om rollen är giltig (Customer eller Admin)

IF @NewRole NOT IN ('Customer', 'Admin')

BEGIN

PRINT 'Ogiltig roll. Tillåtna roller är Customer eller Admin.';

RETURN;

END

-- Uppdatera användarens roll

UPDATE Users

SET Role = @NewRole, UpdatedAt = GETDATE()

WHERE EmailAddress = @EmailAddress;

PRINT 'Användarens roll har uppdaterats!';

END;

--2 Procedur för att skicka verifieringskod

GO

CREATE OR ALTER PROCEDURE VerifyAccount

@EmailAddress NVARCHAR(255)

AS

BEGIN

-- Kontrollera om användaren finns

IF NOT EXISTS (SELECT 1 FROM Users WHERE EmailAddress = @EmailAddress)

BEGIN

PRINT 'E-postadressen hittades inte.';

RETURN;

END

-- Uppdatera IsVerified till 1

UPDATE Users

SET IsVerified = 1

WHERE EmailAddress = @EmailAddress;

PRINT 'Kontot har verifierats.';

END;

--Lösenordsåterställning

GO

CREATE OR ALTER PROCEDURE ForgotPassword

@EmailAddress NVARCHAR(255)

AS

BEGIN

-- Kontrollera om användaren finns

IF NOT EXISTS (SELECT 1 FROM Users WHERE EmailAddress = @EmailAddress)

BEGIN

PRINT 'E-postadressen hittades inte.';

RETURN;

END

-- Skapa en unik reset-kod och sätt giltighetstiden

DECLARE @ResetCode NVARCHAR(255) = NEWID();

DECLARE @Expiry DATETIME = DATEADD(HOUR, 24, GETDATE());

-- Uppdatera användarens reset-kod och giltighetstid

UPDATE Users

SET PasswordResetCode = @ResetCode, PasswordResetExpiry = @Expiry

WHERE EmailAddress = @EmailAddress;

PRINT 'En återställningskod har skickats till användaren.';

END;

--Lägga in nytt glömt lösenord

GO

CREATE OR ALTER PROCEDURE SetForgottenPassword

@Email NVARCHAR(255),

@NewPassword NVARCHAR(255),

@ResetCode NVARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

-- Check if the reset code exists and is not expired

DECLARE @UserID INT;

SELECT @UserID = UserID

FROM Users

WHERE PasswordResetCode = @ResetCode

AND PasswordResetExpiry >= GETDATE();

IF @UserID IS NULL

BEGIN

RAISERROR('Invalid or expired reset code.', 16, 1);

RETURN;

END

-- Update the user's password

UPDATE Users

SET PasswordHash = @NewPassword

WHERE UserID = @UserID;

-- Clear the reset code and expiry from the Users table

UPDATE Users

SET PasswordResetCode = NULL,

PasswordResetExpiry = NULL

WHERE UserID = @UserID;

PRINT 'Password successfully updated.';

END;

--Logga in försök och hantering av låsta konton

GO

CREATE OR ALTER PROCEDURE TryLogin

@EmailAddress NVARCHAR(255),

@Password NVARCHAR(50),

@IPAddress NVARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

-- Skapa temporär tabell om den inte finns

IF OBJECT\_ID('tempdb..##TempLoginLogs') IS NULL

BEGIN

CREATE TABLE ##TempLoginLogs (

AttemptID INT IDENTITY(1,1),

Email NVARCHAR(255),

IPAddress NVARCHAR(50),

AttemptDate DATETIME DEFAULT GETDATE(),

Success BIT

);

END;

-- Hämta användaren

DECLARE @UserID INT, @Salt NVARCHAR(50), @PasswordHash NVARCHAR(128), @IsLockedOut BIT;

SELECT @UserID = UserID, @Salt = Salt, @PasswordHash = PasswordHash, @IsLockedOut = IsLockedOut

FROM Users WHERE EmailAddress = @EmailAddress;

-- Kontrollera om användaren finns

IF @UserID IS NULL

BEGIN

INSERT INTO LoginAttempts (Email, IPAddress, Success, AttemptDate)

VALUES (@EmailAddress, @IPAddress, 0, GETDATE());

INSERT INTO ##TempLoginLogs (Email, IPAddress, Success)

VALUES (@EmailAddress, @IPAddress, 0);

PRINT 'Fel e-postadress eller lösenord.';

RETURN;

END;

-- Kontrollera om kontot är låst

IF @IsLockedOut = 1

BEGIN

PRINT 'Kontot är låst. Kontakta support.';

RETURN;

END;

-- Beräkna hash av det angivna lösenordet + salt

DECLARE @InputHash NVARCHAR(MAX) = CONVERT(NVARCHAR(MAX), HASHBYTES('SHA2\_256', @Password + @Salt), 1);

-- Jämför den beräknade hashen med lagrad hash

IF @InputHash = @PasswordHash

BEGIN

-- Logga lyckat försök

INSERT INTO LoginAttempts (UserID, Email, IPAddress, Success, AttemptDate)

VALUES (@UserID, @EmailAddress, @IPAddress, 1, GETDATE());

-- Spara också i den temporära tabellen

INSERT INTO ##TempLoginLogs (Email, IPAddress, Success)

VALUES (@EmailAddress, @IPAddress, 1);

PRINT 'Inloggningen lyckades!';

END

ELSE

BEGIN

-- Logga misslyckat försök

INSERT INTO LoginAttempts (UserID, Email, IPAddress, Success, AttemptDate)

VALUES (@UserID, @EmailAddress, @IPAddress, 0, GETDATE());

INSERT INTO ##TempLoginLogs (Email, IPAddress, Success)

VALUES (@EmailAddress, @IPAddress, 0);

-- Kontrollera misslyckade försök inom de senaste 15 minuterna

DECLARE @FailedAttempts INT;

SELECT @FailedAttempts = COUNT(\*)

FROM LoginAttempts

WHERE UserID = @UserID

AND Success = 0

AND AttemptDate > DATEADD(MINUTE, -15, GETDATE());

-- Om 3 misslyckade försök blir kontot låst

IF @FailedAttempts >= 3

BEGIN

UPDATE Users SET IsLockedOut = 1 WHERE UserID = @UserID;

PRINT 'Kontot har låsts efter tre misslyckade försök. Vänta 15 minuter eller kontakta support.';

END

ELSE

BEGIN

PRINT 'Inloggningen misslyckades. Försök igen.';

END

END;

END;

GO

CREATE OR ALTER PROCEDURE LockUnlockUser

@EmailAddress NVARCHAR(255),

@Action NVARCHAR(10) -- 'LOCK' eller 'UNLOCK'

AS

BEGIN

IF @Action = 'LOCK'

BEGIN

UPDATE Users

SET IsLockedOut = 1

WHERE EmailAddress = @EmailAddress;

PRINT 'Användaren har blivit låst.';

END

ELSE IF @Action = 'UNLOCK'

BEGIN

UPDATE Users

SET IsLockedOut = 0

WHERE EmailAddress = @EmailAddress;

PRINT 'Användaren har blivit upplåst.';

END

ELSE

BEGIN

PRINT 'Ogiltig åtgärd. Använd LOCK eller UNLOCK.';

END;

END;

--Rapporter (Views med CTE)

--Rapport: Senaste lyckade och misslyckade inloggningar

-- Skapa eller ersätt VIEW

GO

CREATE OR ALTER VIEW UserLoginReport AS

WITH LatestLogins AS (

SELECT

UserID,

MAX(CASE WHEN Success = 1 THEN AttemptDate END) AS LastSuccessfulLogin,

MAX(CASE WHEN Success = 0 THEN AttemptDate END) AS LastFailedLogin

FROM LoginAttempts

GROUP BY UserID

)

SELECT

u.EmailAddress,

u.FirstName,

u.LastName,

ll.LastSuccessfulLogin,

ll.LastFailedLogin

FROM Users u

LEFT JOIN LatestLogins ll ON u.UserID = ll.UserID;

GO

--Rapport: Antal lyckade och misslyckade försök per IP-adress

GO

CREATE OR ALTER VIEW LoginAttemptsPerIP AS

WITH AttemptSummary AS (

SELECT

IPAddress,

COUNT(\*) OVER (PARTITION BY IPAddress) AS TotalAttempts,

SUM(CASE WHEN Success = 1 THEN 1 ELSE 0 END) OVER (PARTITION BY IPAddress) AS SuccessfulAttempts,

SUM(CASE WHEN Success = 0 THEN 1 ELSE 0 END) OVER (PARTITION BY IPAddress) AS FailedAttempts,

AVG(CASE WHEN Success = 1 THEN 1.0 ELSE 0 END) OVER (PARTITION BY IPAddress) AS AvgSuccessfulAttempts,

AttemptDate

FROM LoginAttempts

)

SELECT

IPAddress,

TotalAttempts,

SuccessfulAttempts,

FailedAttempts,

AvgSuccessfulAttempts,

AttemptDate

FROM AttemptSummary

GO

DECLARE @Run BIT = 0; -- Sätt till 1 för att köra frågor och svar, 0 för att hoppa över.

DECLARE @Fråga1 BIT = 0;

DECLARE @Fråga2 BIT = 0;

DECLARE @Fråga3 BIT = 0;

DECLARE @Fråga4 BIT = 0;

DECLARE @Fråga5 BIT = 0;

DECLARE @Fråga6 BIT = 0;

DECLARE @Fråga7 BIT = 0;

DECLARE @Fråga8 BIT = 0;

DECLARE @Fråga9 BIT = 0;

DECLARE @VGFråga1 BIT = 0;

DECLARE @VGFråga2 BIT = 0;

DECLARE @VGFråga3 BIT = 0;

DECLARE @VGFråga4 BIT = 0;

DECLARE @VGFråga5 BIT = 0;

IF @Run = 1

BEGIN

PRINT 'Kör Frågorna och Svaren.';

-----------------------------------------

-- 1. Registera dig som ny användare

IF @Fråga1 = 0

BEGIN

EXEC RegisterUser

@FirstName = 'Georgos',

@LastName = 'Merza',

@EmailAddress = 'georgos.merza@hotmail.com',

@Password = 'Georgos123!',

@Address = 'NackademinGatan 5',

@PostalCode = '12345',

@City = 'Stockholm',

@Country = 'Sweden',

@PhoneNumber = '0701234567';

END

-- 2. Verifiera sitt konto

IF @Fråga2 = 0

BEGIN

EXEC VerifyAccount @EmailAddress = 'georgos.merza@hotmail.com';

END

-- Testa Logga in:

EXEC TryLogin

@EmailAddress = 'georgos.merza@hotmail.com',

@Password = 'Georgos123!',

@IPAddress = '192.168.0.2'

-- 3. Glömt lösenord

IF @Fråga3 = 0

BEGIN

EXEC ForgotPassword @EmailAddress = 'georgos.merza@hotmail.com';

--(Skickas kod till column 'PassWordResetCode')

END

-- Skriv in nytt lösenord

IF @Fråga3 = 0

BEGIN

EXEC SetForgottenPassword

@Email = 'georgos.merza@hotmail.com',

@NewPassword = 'Georgos12345!',

@ResetCode = 'XXXXX-XXXXX-XXXXX'

END

--SELECT \* FROM Users

-- 4.Customer - default, ändra roll till Admin;

IF @Fråga4 = 0

BEGIN

EXEC UpdateUserRole

@EmailAddress = 'georgos.merza@hotmail.com',

@NewRole = 'Admin'

END

-- 5.Låsa en användare;

IF @Fråga5 = 0

BEGIN

EXEC LockUnlockUser

@EmailAddress = 'georgos.merza@hotmail.com',

@Action = 'LOCK';

END

-- 6. Inloggningsförsök i tabell (IP-address,email,date and time, success)

IF @Fråga6 = 0

BEGIN

SELECT \*

FROM LoginAttempts

END

-- 7. Senaste lyckade/misslyckade inloggning (Datum/tid), VIEW

IF @Fråga7 = 0

BEGIN

SELECT \*

FROM UserLoginReport

ORDER BY LastSuccessfulLogin DESC, LastFailedLogin DESC;

END

-- 8. Rapport inloggningsförsök lyckades/misslyckades per ip-adress.

--(visa antal försök total,lyckade,misslyckade,genomsnittliga lyckade försök)

IF @Fråga8 = 0

BEGIN

SELECT \*

FROM LoginAttemptsPerIP

ORDER BY TotalAttempts DESC;

END

-- 9. Manuellt lägg in värden i user- och logg-tabellen

IF @Fråga9 = 0

BEGIN

INSERT INTO LoginAttempts (UserID, Email, IPAddress, Success)

VALUES

(1, 'georgos.merza@hotmail.com', '192.168.0.1', 1),

(1, 'georgos.merza@hotmail.com', '192.168.0.2', 0),

(1, 'georgos.merza@hotmail.com', '192.168.0.1', 0);

END

-- 10. Ren och läsbar kod som följer best practices. (OK)

-- 11. Scriptet ska kunna exekveras UTAN FEL.

--Testa noggrant innan ni lämnar in att inga run time errors eller logiska fel inträffar. (OK)

----------------------------

--FÖR VÄL GODKÄNT

--1. Om man misslyckats med att logga in tre gånger de senaste 15 minuterna kommer man inte in oavsett om man skriver rätt lösenord eller inte. (OK)

IF @VGFråga1 = 0

BEGIN

EXEC TryLogin @EmailAddress = 'georgos.merza@hotmail.com', @Password = 'Fel123!', @IPAddress = '192.168.0.2';

EXEC TryLogin @EmailAddress = 'georgos.merza@hotmail.com', @Password = 'Fel123!', @IPAddress = '192.168.0.2';

EXEC TryLogin @EmailAddress = 'georgos.merza@hotmail.com', @Password = 'Fel123!', @IPAddress = '192.168.0.2';

END;

--2. Skapa stored procedures för trylogin(email,password. ipaddress) och denna ska kontrollera alla saker ovan enl. G och VG krav SAMT returnera lämpliga felkoder ifall det inte gick.

--Kom ihåg att lagra i loggtabellen. Loggar ska även sparas i en temporär tabell i samma SP, som ni ex. kan använda för testning.

--(OK)

IF @VGFråga2 = 0

BEGIN

SELECT \* FROM ##TempLoginLogs

END;

--3. Skapa stored procedure för forgotpassword(email) och denna ska skapa och lagra password koden (token). Eventuella felkoder ska returneras.

IF @VGFråga3 = 0

BEGIN

EXEC ForgotPassword @EmailAddress = 'georgos.merza@hotmail.com';

END;

--4. Gör stored procedure för setforgottenpassword(email,password,token). Kolla så rätt token, rätt tid (ej expired). Eventuella felkoder ska returneras.

IF @VGFråga4 = 0

BEGIN

EXEC SetForgottenPassword

@Email = 'georgos.merza@hotmail.com',

@NewPassword = 'NyttLösenord123!',

@ResetCode = 'XXXXX-XXXXX-XXXXX';

END;

ELSE BEGIN

PRINT 'Frågorna och svaren körs inte.';

END;

END

--5. Demonstrera användandet av era SP. Dessa ska returnera resultat som ni visar:

--Samtliga SP och svar på frågorna finns uppe i ordning enligt inlämningsuppgiften

--6. Ni ska konstruera scriptet och rapporterna med optimering i åtanke. Detta ska tydligt redovisas i er dokumentation. (OK)

--Ni ska redovisa hur ni har arbetat med optimering, hur systemet skulle kunna förbättras i framtiden i relation till optimering,

--vad man kan göra för att testa prestandan och ev. annat som ni tycker är relevant.

--(OK)