Backup queries:

ALTER TABLE MasterTable

ADD CONSTRAINT fk\_xyz

FOREIGN KEY (xyz) REFERENCES ChildTable (xyz) ON DELETE CASCADE

Queries:

CREATE TABLE [dbo].[USERS] (

[UserID] INT IDENTITY (1, 1) NOT NULL,

[UserPassword] VARCHAR (MAX) NOT NULL,

[UserName] VARCHAR (30) NOT NULL,

[RealUserName] VARCHAR (MAX) NOT NULL,

[ProfilePicture] VARBINARY (MAX) NULL,

[TagLine] VARCHAR (200) NULL,

PRIMARY KEY CLUSTERED ([UserID] ASC),

CONSTRAINT [UK\_UserName] UNIQUE NONCLUSTERED ([UserName] ASC)

);

CREATE TABLE POST

(

PostID INT IDENTITY( 1 , 1) PRIMARY KEY,

UserID INT FOREIGN KEY REFERENCES USERS (UserID),

PostDesc VARCHAR(255),

BookMarked BIT NOT NULL,

Likes INT NOT NULL DEFAULT 0,

Location VARCHAR(30),

Image VARBINARY(MAX),

Video VARBINARY(MAX),

TimeLine TIMESTAMP NOT NULL

)

---- Below can’t be used right away, needs some research

CREATE VIEW SEARCH

AS

SELECT [UserID], [UserName], [RealUserName]

FROM [USERS]

WHERE [UserName] = ???? OR [UserName] = ????

ORDERBY [USERID]

CREATE TABLE STORY

(

StoryID INT IDENTITY(1,1) PRIMARY KEY,

Image VARBINARY(MAX),

Video VARBINARY(MAX),

PublishDate TIMESTAMP NOT NULL,

);

CREATE TABLE COMMENT

(

CommentID INT IDENTITY(1,1) PRIMARY KEY,

PostID INT FOREIGN KEY REFERENCES POST (PostID),

UserID INT FOREIGN KEY REFERENCES USERS (UserID)

);

CREATE TABLE CHAT

(

ChatID INT IDENTITY(1,1) PRIMARY KEY,

UserID INT FOREIGN KEY REFERENCES USERS(UserID),

ChatData VARCHAR(255)

);

CREATE TABLE Message

(

MessageID INT IDENTITY(1,1) PRIMARY KEY,

ChatID INT FOREIGN KEY REFERENCES CHAT(ChatID),

MessageText VARCHAR(255) NOT NULL,

);

CREATE TABLE FOLLOWER

(

FollowerID INT IDENTITY(1,1) PRIMARY KEY,

UserID INT FOREIGN KEY REFERENCES USERS(UserID)

);

CREATE TABLE FOLLOWING

(

FollowingID INT IDENTITY(1,1) PRIMARY KEY,

UserID INT FOREIGN KEY REFERENCES USERS(UserID)

);

--- NO NEED FOR CHAT SECTION

CREATE TABLE USER\_LIKE

(

LikeID INT IDENTITY(1,1) PRIMARY KEY,

PostID INT FOREIGN KEY REFERENCES POST(PostID),

UserID INT FOREIGN KEY REFERENCES USERS(UserID),

);

--- Problem with the implementation of like feed need to discuss further

CREATE VIEW LIKE\_FEED

AS

SELECT [PostID]

FROM USER\_LIKE AS UL

INNER JOIN POST AS P

ON P.UserID = UL.UserID AND P.PostID = UL.PostID

CREATE VIEW BOOKMARKS

AS

SELECT [UserID], [PostID]

FROM POST

WHERE BookMarked = 1

--- Currently not considering to make a database table for signup/login as we just simply

Need to add the user or login with the user which can be done directly with the user table and from program directly

CREATE PROCEDURE Add\_User

@UserName VARCHAR(20),

@RealUserName VARCHAR(30),

@UserPassword VARCHAR(20),

@Picture VARBINARY(MAX) = NULL,

@Tagline VARCHAR(200) = NULL

AS

if (@Picture IS NULL)

BEGIN

INSERT INTO USERS(UserName, RealUserName, UserPassword, TagLine) VALUES(@UserName, @RealUserName, @UserPassword, @Tagline)

END

if (@Tagline IS NULL)

BEGIN

INSERT INTO USERS(UserName, RealUserName, UserPassword, ProfilePicture) VALUES(@UserName, @RealUserName, @UserPassword, @Picture)

END

CREATE FUNCTION Check\_UserName\_And\_Password(

@UserName VARCHAR(20),

@Password VARCHAR(20)

)

RETURNS bit

AS

BEGIN

RETURN

(

SELECT ISNULL

(

(

SELECT 1

From USERS

WHERE UserName = @UserName AND UserPassword = @Password

), 0

)

)

END

CREATE FUNCTION Check\_UserName(

@UserName VARCHAR(20)

)

RETURNS bit

AS

BEGIN

RETURN

(

SELECT ISNULL

(

(

SELECT 1

From USERS

WHERE UserName = @UserName

), 0

)

)

END

CREATE FUNCTION Check\_Password(

@Password VARCHAR(20)

)

RETURNS bit

AS

BEGIN

RETURN

(

SELECT ISNULL

(

(

SELECT 1

From USERS

WHERE UserPassword = @Password

), 0

)

)

END

CREATE FUNCTION Check\_Password(

@UserName VARCHAR(20),

@Password VARCHAR(20)

)

RETURNS bit

AS

BEGIN

RETURN

(

SELECT ISNULL

(

(

SELECT 1

From USERS

WHERE UserName = @UserName AND UserPassword = @Password

), 0

)

)

END

CREATE PROCEDURE Build\_User\_Posts\_Table

@UserID VARCHAR(MAX),

@UserName VARCHAR(MAX)

AS

DECLARE @SQLCmd VARCHAR(MAX)

DECLARE @TableName VARCHAR(MAX)

SET @TableName = @UserName + '\_' + @UserID + '\_PostTable'

SET @SQLCmd = 'CREATE TABLE '+ @TableName + ' (PostID INT IDENTITY( 1 , 1) PRIMARY KEY, PostDesc VARCHAR(255), BookMarked BIT NOT NULL,Likes INT NOT NULL DEFAULT 0, Location VARCHAR(30), Image VARBINARY(MAX), Video VARBINARY(MAX), TimeLine TIMESTAMP DEFAULT GETDATE()'

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = @TableName)

BEGIN

EXEC (@SQLCmd)

END

GO

**CREATE PROCEDURE Add\_Post**

**(**

**@UserName VARCHAR(MAX),**

**@UserID VARCHAR(MAX),**

**@PostDesc VARCHAR(MAX),**

**@Location VARCHAR(MAX),**

**@Image VARBINARY NULL**

**)**

**AS**

**DECLARE @SQLCmd VARCHAR(MAX)**

**DECLARE @TimeLine VARCHAR(MAX)**

**DECLARE @BookMarked BIT,**

**IF @PostDesc IS NULL**

**BEGIN**

**SET @PostDesc = '';**

**END**

**IF @Location IS NULL**

**BEGIN**

**SET @Location = '';**

**END**

**SET @BookMarked = 0;**

**SET @TimeLine = CONVERT(VARCHAR(255), GETDATE());**

**SET @SQLCmd = 'INSERT INTO ' + @UserName + '\_' + @UserID + '\_POSTTable' + '( PostDesc, BookMarked, Location, Likes, Image, TimeLine) VALUES( ' + @PostDesc + ',' + CONVERT(VARCHAR(2), @BookMarked) + ',' + @Location + ', 0,' + CONVERT(VARCHAR(MAX), @Image) + ',' + CONVERT(VARCHAR(MAX), @TimeLine) + ')'**

**EXECUTE (@SQLCmd)**

**GO**

CREATE PROCEDURE Build\_User\_Stories\_Table

@UserID VARCHAR(MAX),

@UserName VARCHAR(MAX)

AS

DECLARE @SQLCmd VARCHAR(MAX)

DECLARE @TableName VARCHAR(MAX)

SET @TableName = @UserName + '\_' + @UserID + '\_StoryTable'

SET @SQLCmd = 'CREATE TABLE '+ @TableName + ' (StoryID INT IDENTITY( 1 , 1) PRIMARY KEY, Image VARBINARY(MAX), Video VARBINARY(MAX), PostTime datetime DEFAULT CURRENT\_TIMESTAMP, DueDate datetime NOT NULL)'

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = @TableName)

BEGIN

EXEC (@SQLCmd)

END

CREATE PROCEDURE Delete\_Post(

@PostID INT,

@UserID INT,

@UserName VARCHAR(MAX)

)

AS

DECLARE @SQLCmd VARCHAR(MAX)

SET @SQLCmd = 'DELETE FROM ' + @UserName + '\_' + @UserID + '\_PostTable' +'WHERE PostID = ' + @PostID

EXECUTE @SQLCmd

GO

CREATE PROCEDURE Delete\_Story(

@StoryID INT,

@UserID INT,

@UserName VARCHAR(MAX)

)

AS

DECLARE @SQLCmd VARCHAR(MAX)

SET @SQLCmd = 'DELETE FROM ' + @UserName + '\_' + @UserID + '\_StoryTable' +'WHERE StoryID = ' + @StoryID

EXECUTE @SQLCmd

GO

DECLARE @UserName VARCHAR(MAX)

DECLARE @UserID VARCHAR(MAX)

DECLARE @TableName VARCHAR(MAX)

DECLARE @ViewName VARCHAR(MAX)

DECLARE @SQLCmd VARCHAR(MAX)

SET @UserName = 'Eqan'

SET @UserID = '3'

SET @TableName = @UserName + '\_' + @UserID + '\_StoryTable'

SET @ViewName = @UserName + '\_' + @UserID + '\_StoryViewTable'

SET @SQLCmd = 'DROP VIEW ' + @ViewName

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.VIEWS WHERE TABLE\_NAME = @ViewName)

BEGIN

EXEC (@SQLCmd)

END

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.VIEWS WHERE TABLE\_NAME = @ViewName)

BEGIN

SET @SQLCmd = 'CREATE VIEW ' + @ViewName + ' AS SELECT \* FROM ' + @TableName + ' WHERE DueDate >= CURRENT\_TIMESTAMP'

END

EXEC @SQLCmd

GO

CREATE PROCEDURE Build\_User\_Followers\_Table

(

@UserID VARCHAR(MAX),

@UserName VARCHAR(MAX)

)

AS

DECLARE @TableName VARCHAR(MAX)

DECLARE @SQLCmd VARCHAR(MAX)

SET @TableName = @UserName + '\_' + @UserID + '\_FollowersTable'

SET @SQLCmd = 'CREATE TABLE ' + @TableName + ' (FollowersID INT IDENTITY(1,1) PRIMARY KEY, UserID INT, UserName VARCHAR(30))'

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = @TableName)

BEGIN

EXEC (@SQLCmd)

END

GO

CREATE PROCEDURE Build\_User\_Following\_Table

(

@UserID VARCHAR(MAX),

@UserName VARCHAR(MAX)

)

AS

DECLARE @TableName VARCHAR(MAX)

DECLARE @SQLCmd VARCHAR(MAX)

SET @TableName = @UserName + '\_' + @UserID + '\_FollowingTable'

SET @SQLCmd = 'CREATE TABLE ' + @TableName + ' (FollowingID INT IDENTITY(1,1) PRIMARY KEY, UserID INT, UserName VARCHAR(30))'

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = @TableName)

BEGIN

EXEC (@SQLCmd)

END

GO

CREATE PROCEDURE Add\_Following

(

@UserID VARCHAR(MAX),

@UserName VARCHAR(MAX),

@FollowingID VARCHAR(MAX),

@FollowingName VARCHAR(MAX)

)

AS

DECLARE @SQLCmd VARCHAR(MAX)

SET @SQLCmd = 'INSERT INTO ' + @UserName + '\_' + @UserID + '\_FollowingTable' + ' ( UserID, UserName) VALUES(' + @FollowingID + ', ' + @FollowingName + ')'

EXEC @SQLCmd

GO

CREATE PROCEDURE Add\_Follower

(

@UserID VARCHAR(MAX),

@UserName VARCHAR(MAX),

@FollowerID VARCHAR(MAX),

@FollowerName VARCHAR(MAX)

)

AS

DECLARE @SQLCmd VARCHAR(MAX)

SET @SQLCmd = 'INSERT INTO ' + @UserName + '\_' + @UserID + '\_FollowersTable' + ' ( UserID, UserName) VALUES(' + @FollowerID + ', ' + @FollowerName + ')'

EXEC @SQLCmd

CREATE PROCEDURE Build\_View\_For\_All\_Following\_Posts

AS

DECLARE @TableName VARCHAR(MAX)

DECLARE @SQLCmd VARCHAR(MAX)

SET @TableName = 'View\_For\_All\_Following\_Posts'

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.VIEWS WHERE TABLE\_NAME = @TableName)

BEGIN

SET @SQLCmd = 'DROP VIEW ' + @TableName

EXEC (@SQLCmd)

END

SET @SQLCmd = 'CREATE VIEW ' + @TableName + ' AS SELECT NULL AS UserID INT, UserName VARCHAR(MAX) , PostDesc VARCHAR(255), BookMarked BIT NOT NULL,Likes INT NOT NULL DEFAULT 0, Location VARCHAR(30), Image VARBINARY(MAX), Video VARBINARY(MAX), TimeLine datetime DEFAULT CURRENT\_TIMESTAMP)'

EXEC (@SQLCmd)

CREATE PROCEDURE View\_For\_All\_Following\_Posts

(

@UserID VARCHAR(MAX),

@UserName VARCHAR(MAX)

)

AS

DECLARE @TableName VARCHAR(MAX)

DECLARE @SQLCmd VARCHAR(MAX)

SET @TableName = @UserName + '\_' + @UserID + '\_PostTable'

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.VIEWS WHERE TABLE\_NAME = @TableName)

BEGIN

SET @SQLCmd = 'CREATE VIEW View\_For\_All\_Posts AS SELECT ' + @userID + ' AS UserID,'+ @UserName +' AS UserName,PostDesc, BookMarked, Likes, Location, Image, Video, TimeLine FROM ' + @TableName

EXEC (@SQLCmd)

END

ELSE

BEGIN

SET @SQLCmd = 'INSERT INTO View\_For\_All\_Posts SELECT \* FROM ' + @TableName

EXEC (@SQLCmd)

END