```
use sp21_3342_tun49199;
/*
    Description: These SQL scripts create the items used
    by the Email WebApp within an existing database.
    The scripts include table creation, constraints,
    stored procuedures, and also a starter set of sample
    data. Note: the scripts work with an existing database.
    Update the USE statement above to name the database
    you are connected to in order to run the scripts.

    Changes Made:
    tun49199 - 2021-03-03 - Original code.


    Testing Scripts:

    DECLARE @GetDate DATETIME
    SET @GetDate = GETDATE();
    DECLARE @RecvEmailList VARCHAR(4094)
    --SET @RecvEmailList = 'jims@temple.edu;jims-
      admin@temple.edu;professor@temple.edu'
    SET @RecvEmailList = 'professor@temple.edu'

    exec dbo.Email_Insert_SP
        @SendEmailAddress = 'jims@temple.edu',
        @RecvEmailList = @RecvEmailList,
        @EmailSubject = 'henlo?',
        @EmailBody = 'a man a plan a canal panama',
        @DateTimeStamp = @GetDate



    select * from dbo.Account
    select * from dbo.Email
    select * from dbo.EmailReceipt join dbo.Tags on tags.Tagid = emailreceipt.tagid
      order by emailreceipt.emailid

    select * from dbo.Account join dbo.Tags on Tags.AccountId = Account.AccountId
      where Account.CreatedEmailAddress = 'jims@temple.edu'

    select Email.*
    from Account
    join AccountRole on AccountRole.AccountId = Account.AccountId
    join Email on Email.SendAccountId = Account.AccountId

    select Email.*,EmailReceipt.*
    from Account
    join AccountRole on AccountRole.AccountId = Account.AccountId
    join Email on Email.SendAccountId = Account.AccountId
    join EmailReceipt on EmailReceipt.EmailId = Email.EmailId
    join Tags on Tags.TagId = EmailReceipt.TagId
```

```
*/
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/*****************************************************************************
 *    Drop constrints to be recreated below
 *****************************************************************************/
    IF (SELECT object_id('Email_Account_FK')) is not null
    ALTER TABLE dbo.Email DROP CONSTRAINT Email_Account_FK;

    IF (SELECT object_id('EmailReceipt_PK')) is not null
    ALTER TABLE dbo.EmailReceipt DROP CONSTRAINT EmailReceipt_PK;

    IF (SELECT object_id('EmailReceipt_Email_FK')) is not null
    ALTER TABLE dbo.EmailReceipt DROP CONSTRAINT EmailReceipt_Email_FK;

    IF (SELECT object_id('EmailReceipt_Tags_FK')) is not null
    ALTER TABLE dbo.EmailReceipt DROP CONSTRAINT EmailReceipt_Tags_FK;

    IF (SELECT object_id('Tags_PK')) is not null
    ALTER TABLE dbo.Tags DROP CONSTRAINT Tags_PK;

    IF (SELECT object_id('Tags_Account_FK')) is not null
    ALTER TABLE dbo.Tags DROP CONSTRAINT Tags_Account_FK;

    IF (SELECT object_id('AccountRole_PK')) is not null
    ALTER TABLE dbo.AccountRole DROP CONSTRAINT AccountRole_PK;

    IF (SELECT object_id('Account_PK')) is not null
    ALTER TABLE dbo.Account DROP CONSTRAINT Account_PK;

    IF (SELECT object_id('SecurityQuestion_PK')) is not null
    ALTER TABLE dbo.SecurityQuestion DROP CONSTRAINT SecurityQuestion_PK;

    IF (SELECT object_id('Email_PK')) is not null
    ALTER TABLE dbo.Email DROP CONSTRAINT Email_PK;

/*****************************************************************************
 *    Drop tables to be recreated with starter data
 *****************************************************************************/
    IF (SELECT object_id('dbo.Account')) is not null
    DROP TABLE dbo.Account;

    IF (SELECT object_id('dbo.AccountRole')) is not null
    DROP TABLE dbo.AccountRole;

    IF (SELECT object_id('dbo.SecurityQuestion')) is not null
    DROP TABLE dbo.SecurityQuestion;

    IF (SELECT object_id('dbo.Email')) is not null
    DROP TABLE dbo.Email;
```

```
    IF (SELECT object_id('dbo.EmailReceipt')) is not null
    DROP TABLE dbo.EmailReceipt;

    IF (SELECT object_id('dbo.Tags')) is not null
    DROP TABLE dbo.Tags;

/*************************************************************************
 *    Create AccountRole table
 *************************************************************************/
    CREATE TABLE dbo.AccountRole (
        AccountRoleId INT IDENTITY(1,1),
        AccountId INT,
        AccountRoleType VARCHAR(14) NOT NULL, --User or Administrator
        DateTimeStamp DATETIME DEFAULT GETDATE(),
        CONSTRAINT AccountRole_PK PRIMARY KEY CLUSTERED (AccountRoleId)
    );
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*************************************************************************
 *    Create Account table
 *************************************************************************/
    CREATE TABLE dbo.Account (
        AccountId INT IDENTITY(1,1),
        UserName VARCHAR(50),
        UserAddress VARCHAR(254),
        PhoneNumber VARCHAR(50),
        CreatedEmailAddress VARCHAR(254) UNIQUE,
        ContactEmailAddress VARCHAR(254),
        Avatar INT,
        AccountPassword VARBINARY(MAX),
        Active VARCHAR(5),
        DateTimeStamp DATETIME DEFAULT GETDATE()
        CONSTRAINT Account_PK PRIMARY KEY CLUSTERED (AccountId)
    );
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*************************************************************
 *    Create SecutityQuestion table
 *
 *    QuestionType allowable values
 *        City   'In what town or city was your first full time job?'
 *        Phone  'What were the last four digits of your childhood telephone number?'
 *        School 'What primary school did you attend?'
 *************************************************************/
    CREATE TABLE dbo.SecurityQuestion (
```

```
        SecurityQuestionId INT IDENTITY(1,1),
        AccountId INT NOT NULL,
        Question VARCHAR(254) NOT NULL,
        QuestionType VARCHAR(14) NOT NULL,
        Response VARCHAR(254),
        DateTimeStamp DATETIME DEFAULT GETDATE(),

        CONSTRAINT SecurityQuestion_PK PRIMARY KEY CLUSTERED (SecurityQuestionId)
    );
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*************************************************************************
 *    Create Email table
 *************************************************************************/
    CREATE TABLE dbo.Email (
        EmailId INT IDENTITY(1,1),
        AccountId INT,
        RecvEmailList VARCHAR(4094),
        EmailSubject VARCHAR(254),
        EmailBody VARCHAR(MAX),
        DateTimeStamp DATETIME DEFAULT GETDATE(),

        CONSTRAINT Email_PK PRIMARY KEY (EmailId),
    );
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*************************************************************************
 *    Create Tags table
 *************************************************************************/
    CREATE TABLE dbo.Tags (
        TagId INT IDENTITY(1,1),
        TagName VARCHAR(254),
        TagType VARCHAR(6),
        AccountId INT,
        DateTimeStamp DATETIME DEFAULT GETDATE(),
        CONSTRAINT Tags_PK PRIMARY KEY CLUSTERED (TagId),

        CONSTRAINT Tags_Account_FK FOREIGN KEY (AccountId)
        REFERENCES dbo.Account(AccountId)
    );
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*************************************************************************
```

```
 *    Create EmailReceipt table
 ******************************************************************************/
    CREATE TABLE dbo.EmailReceipt (
        EmailReceiptId INT IDENTITY(1,1),
        AccountIdSend INT,
        AccountIdRecv INT,
        EmailId INT,
        TagId INT,
        EmailFlag VARCHAR(12),
        DateTimeStamp DATETIME DEFAULT GETDATE(),

        CONSTRAINT EmailReceipt_PK PRIMARY KEY CLUSTERED (EmailReceiptId),

        CONSTRAINT EmailReceipt_Email_FK FOREIGN KEY (EmailId)
        REFERENCES dbo.Email(EmailId),

        CONSTRAINT EmailReceipt_Tags_FK FOREIGN KEY (TagId)
        REFERENCES dbo.Tags(TagId)
    );
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*****************************************************************************
 *    Description: Procedure to INSERT a new account using
 *                 the Account, Tags, and SecurityQuestion tables.
 *                 Each user gets a default set of tags/folders.
 *                 Also the user must responsd to security questions to
 *                 help verify their identity if in the future they
 *                 need to reset their password.
 *****************************************************************************/
    IF (SELECT object_id('dbo.Account_Insert_SP')) is not null
    DROP PROCEDURE dbo.Account_Insert_SP;
    GO

    CREATE PROCEDURE dbo.Account_Insert_SP
        @UserName VARCHAR(50)='',
        @UserAddress VARCHAR(254)='',
        @PhoneNumber VARCHAR(50)='',
        @CreatedEmailAddress VARCHAR(254)='',
        @ContactEmailAddress VARCHAR(254)='',
        @Avatar INT=0,
        @AccountPassword VARBINARY(MAX),
        @Active VARCHAR(5)='',
        @ResponseCity VARCHAR(254)= '',
        @ResponsePhone VARCHAR(254)= '',
        @ResponseSchool VARCHAR(254)= '',
        @AccountRoleType VARCHAR(14) = '',
        @DateTimeStamp DATETIME
    AS
    BEGIN TRANSACTION
```

```
        BEGIN
            DECLARE @AccountId INT;
            IF (@DateTimeStamp IS NULL ) SET @DateTimeStamp = GETDATE();

            INSERT INTO dbo.Account (UserName,
                                     UserAddress,
                                     PhoneNumber,
                                     CreatedEmailAddress,
                                     ContactEmailAddress,
                                     Avatar,
                                     AccountPassword,
                                     Active,
                                     DateTimeStamp)
                             VALUES (@UserName,
                                     @UserAddress,
                                     @PhoneNumber,
                                     @CreatedEmailAddress,
                                     @ContactEmailAddress,
                                     @Avatar,
                                     @AccountPassword,
                                     @Active,
                                     @DateTimeStamp);
            IF @@ERROR = -1 GOTO On_Account_Insert_Error

            -- Use the new @@IDENTITY from the new dbo.Account INSERT the Account Role
            SELECT @AccountId = @@IDENTITY;
            INSERT INTO dbo.AccountRole (AccountId,
                                         AccountRoleType,
                                         DateTimeStamp)
                                 VALUES (@AccountId,
                                         @AccountRoleType,
                                         @DateTimeStamp);
            IF @@ROWCOUNT = 0 GOTO On_Account_Insert_Error

            -- All email accounts get a 'model' set of Tags
            -- including 'Inbox'
            --          'Sent'
            --          'Flag'
            --          'Junk'
            --          'Trash'
            -- Tags addeed by the user will have a 'Custom' TagType
            INSERT INTO dbo.Tags (TagName,
                                  TagType,
                                  AccountId,
                                  DateTimeStamp)
                                  VALUES('Inbox',
                                         'Model',
                                         @AccountId,
                                         @DateTimeStamp);
            IF @@ROWCOUNT = 0 GOTO On_Account_Insert_Error

            INSERT INTO dbo.Tags (TagName,
```

```
                        TagType,
                        AccountId,
                        DateTimeStamp)
                        VALUES('Sent',
                        'Model',
                        @AccountId,
                        @DateTimeStamp);
        IF @@ROWCOUNT = 0 GOTO On_Account_Insert_Error

        INSERT INTO dbo.Tags (TagName,
                        TagType,
                        AccountId,
                        DateTimeStamp)
                        VALUES('Flag',
                        'Model',
                        @AccountId,
                        @DateTimeStamp);
        IF @@ROWCOUNT = 0 GOTO On_Account_Insert_Error

        INSERT INTO dbo.Tags (TagName,
                        TagType,
                        AccountId,
                        DateTimeStamp)
                        VALUES('Junk',
                        'Model',
                        @AccountId,
                        @DateTimeStamp);
        IF @@ROWCOUNT = 0 GOTO On_Account_Insert_Error

        INSERT INTO dbo.Tags (TagName,
                        TagType,
                        AccountId,
                        DateTimeStamp)
                        VALUES('Trash',
                        'Model',
                        @AccountId,
                        @DateTimeStamp);
        IF @@ROWCOUNT = 0 GOTO On_Account_Insert_Error

        DECLARE @Question VARCHAR(254);
        SELECT @Question = 'In what town or city was your first full time job?';
        INSERT INTO dbo.SecurityQuestion(AccountId,
                                    QuestionType,
                                    Question,
                                    Response,
                                    DateTimeStamp)
                                VALUES (@AccountId,
                                    'City',
                                    @Question,
                                    @ResponseCity,
                                    @DateTimeStamp);
        IF @@ERROR = -1 GOTO On_Account_Insert_Error
```

```
            SELECT @Question = 'What were the last four digits of your childhood
              telephone number?';
            INSERT INTO dbo.SecurityQuestion(AccountId,
                                             QuestionType,
                                             Question,
                                             Response,
                                             DateTimeStamp)
                                      VALUES (@AccountId,
                                             'Phone',
                                             @Question,
                                             @ResponsePhone,
                                             @DateTimeStamp);
            IF @@ERROR = -1 GOTO On_Account_Insert_Error

            SELECT @Question = 'What primary school did you attend?';
            INSERT INTO dbo.SecurityQuestion(AccountId,
                                             QuestionType,
                                             Question,
                                             Response,
                                             DateTimeStamp)
                                      VALUES (@AccountId,
                                             'School',
                                             @Question,
                                             @ResponseSchool,
                                             @DateTimeStamp);
            IF @@ERROR = -1 GOTO On_Account_Insert_Error

            COMMIT TRANSACTION
        END
        RETURN 0
        On_Account_Insert_Error:
            ROLLBACK TRANSACTION
            RETURN -1
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*****************************************************************
 *    Description: Procedure to answer security questions.
 *    Return the number of correct responses.
 *
 *    Parameters that must be passed in:
 *        @CreatedEmailAddress
 *        @ResponseCity
 *        @ResponsePhone
 *        @ResponseSchool
 *
 *****************************************************************/
    IF (SELECT object_id('dbo.Account_Security_Questions_SP')) is not null DROP
      PROCEDURE dbo.Account_Security_Questions_SP;
```

```
    GO
    CREATE PROCEDURE dbo.Account_Security_Questions_SP
        @CreatedEmailAddress VARCHAR(254)='',
        @ResponseCity VARCHAR(254),
        @ResponsePhone VARCHAR(254),
        @ResponseSchool VARCHAR(254)
    AS
    BEGIN
        DECLARE @MatchCount INT
        SET @MatchCount = 0
        SELECT @MatchCount = @MatchCount + 1
            FROM dbo.Account a
            JOIN dbo.SecurityQuestion s on s.AccountId = a.AccountId
            WHERE a.CreatedEmailAddress = @CreatedEmailAddress
            AND s.Response = @ResponseCity
            AND s.QuestionType = 'City'

        SELECT @MatchCount = @MatchCount + 1
        FROM dbo.Account a
            JOIN dbo.SecurityQuestion s on s.AccountId = a.AccountId
            WHERE a.CreatedEmailAddress = @CreatedEmailAddress
            AND s.Response = @ResponsePhone
            AND s.QuestionType = 'Phone'

        SELECT @MatchCount = @MatchCount + 1
            FROM dbo.Account a
            JOIN dbo.SecurityQuestion s on s.AccountId = a.AccountId
            WHERE a.CreatedEmailAddress = @CreatedEmailAddress
            AND s.Response = @ResponseSchool
            AND s.QuestionType = 'School'

        SELECT @MatchCount as NumOfCorrectResponses
    END
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*************************************************************************
 *    Description: Procedure to LOGIN.
 *
 *    Parameters that must be passed in:
 *        @CreatedEmailAddress
 *        @AccountPassword
 *
 *************************************************************************/
    IF (SELECT object_id('dbo.Account_Login_SP')) is not null
    DROP PROCEDURE dbo.Account_Login_SP;
    GO

    CREATE PROCEDURE dbo.Account_Login_SP
        @CreatedEmailAddress VARCHAR(254),
```

```
        @AccountPassword VARBINARY(MAX)
    AS
    BEGIN
        SELECT
            Account.AccountId,
            Account.UserName,
            Account.UserAddress,
            Account.PhoneNumber,
            Account.CreatedEmailAddress,
            Account.ContactEmailAddress,
            Account.Avatar,
            Account.AccountPassword,
            Account.Active,
            Account.DateTimeStamp,
            AccountRole.AccountRoleType
        FROM dbo.Account
        JOIN dbo.AccountRole ON AccountRole.AccountId = Account.AccountId
        WHERE CreatedEmailAddress = @CreatedEmailAddress
        AND AccountPassword = @AccountPassword;
    END
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
    GO
/*****************************************************************
 *    Description: Procedure to UPDATE password in the Account table.
 *****************************************************************/
    IF (SELECT object_id('dbo.Account_Update_Password_SP')) is not null
    DROP PROCEDURE dbo.Account_Update_Password_SP;
    GO
    CREATE PROCEDURE dbo.Account_Update_Password_SP
        @CreatedEmailAddress VARCHAR(254),
        @AccountPassword VARBINARY(MAX)
    AS
    BEGIN TRANSACTION
        BEGIN
            UPDATE dbo.Account
            SET Account.AccountPassword = @AccountPassword
            WHERE Account.CreatedEmailAddress = @CreatedEmailAddress
            IF @@ERROR = -1 GOTO On_Account_Update_Password_Error

            COMMIT TRANSACTION
        END
        RETURN 0
        On_Account_Update_Password_Error:
            ROLLBACK TRANSACTION
            RETURN -1
    GO
    SET ANSI_NULLS ON
    GO
    SET QUOTED_IDENTIFIER ON
```

```
    GO

/*****************************************************************************
 *    Description: Procedure to INSERT an email.
 *
 *    Parameters that must be passed in:
 *      @SendEmailAddress
 *      @RecvEmailList
 *      @EmailSubject
 *      @EmailBody
 *      @DateTimeStamp
 *
 *****************************************************************************/
    IF (SELECT object_id('dbo.Email_Send_SP')) is not null
    DROP PROCEDURE dbo.Email_Send_SP;
    GO

    CREATE PROCEDURE dbo.Email_Send_SP
        @SendEmailAddress VARCHAR(254)='',
        @RecvEmailList VARCHAR(4094)='', --1 byte/char + 2 bytes for length
        @EmailSubject VARCHAR(254)='',
        @EmailBody VARCHAR(MAX)='',
        @DateTimeStamp DATETIME=''
    AS
    BEGIN TRANSACTION
        BEGIN
            DECLARE @SendAccountId INT
            SELECT @SendAccountId = AccountId
                FROM dbo.Account
                WHERE CreatedEmailAddress = @SendEmailAddress;
            DECLARE @EmailId INT
            DECLARE @TagsSentId INT
            SELECT @TagsSentId = TagId
                FROM dbo.Tags
                JOIN dbo.Account ON Account.AccountId = Tags.AccountId
                                AND Tags.TagName = 'Sent'
                WHERE Account.AccountId = @SendAccountId;

            ----do not send email unless account exists
            --IF NOT EXISTS(SELECT *
            --  FROM dbo.Account
            --  WHERE Account.CreatedEmailAddress = @RecvEmailList)
            --  GOTO On_Email_Send_Error

            ----do not allow email to Administrator type accounts
            --IF EXISTS(SELECT *
            --  FROM dbo.Account
            --  JOIN dbo.AccountRole on AccountRole.AccountId = Account.AccountId
            --  WHERE Account.CreatedEmailAddress = @RecvEmailList
            --  AND AccountRole.AccountRoleType = 'Administrator')
            --  GOTO On_Email_Send_Error
```

```
            --create a list of recv account ids using the
            --semi-colon ';'separated values
            IF (SELECT object_id(N'tempdb..#RecvEmailAccountId')) is not null
                DROP TABLE #RecvEmailAccountId;
            SELECT
                Account.AccountId
            INTO #RecvEmailAccountId
            FROM STRING_SPLIT(@RecvEmailList,';') AS EmailAddress
            JOIN dbo.Account ON Account.CreatedEmailAddress = EmailAddress.value

            INSERT INTO dbo.Email
                    (AccountId,
                    RecvEmailList,
                    EmailSubject,
                    EmailBody)
                VALUES
                    (@SendAccountId,
                    @RecvEmailList,
                    @EmailSubject,
                    @EmailBody)
            IF @@ERROR = -1 GOTO On_Email_Send_Error
            SET @EmailId = @@IDENTITY

            -- insert 'inbox' copy of email into the EmailReceipt table
            -- for each receive email addesses in the ';' separated list
            INSERT INTO EmailReceipt
                    (AccountIdSend,
                    AccountIdRecv,
                    EmailId,
                    TagId,
                    EmailFlag)
                SELECT
                    @SendAccountId,
                    RecvEmail.AccountId,
                    @EmailId,
                    (SELECT TagId
                        FROM dbo.Tags
                        WHERE Tags.AccountId = Account.AccountId
                        AND Tags.TagName = 'Inbox'),
                    'No'
                FROM #RecvEmailAccountId RecvEmail
                JOIN dbo.Account ON Account.AccountId = RecvEmail.AccountId

            IF @@ERROR = -1 GOTO On_Email_Send_Error

            -- insert 'sent' copy of email into the EmailReceipt
            INSERT INTO EmailReceipt
                    (AccountIdSend,
                    AccountIdRecv,
                    EmailId,
                    TagId,
                    EmailFlag)
```

```
                SELECT
                    @SendAccountId,
                    @SendAccountId,
                    @EmailId,
                    @TagsSentId,
                    'No'
            IF @@ERROR = -1 GOTO On_Email_Send_Error

            COMMIT TRANSACTION
        END
        RETURN 0
        On_Email_Send_Error:
            ROLLBACK TRANSACTION
            RETURN -1
    GO
/*************************************************************************
 *    Description: Procedure to SELECT emails for the
 *                 given email address with a certain TagName.
 *
 *    Parameters that must be passed in:
 *        @CreatedEmailAddress
 *        @TagName
 *
 *************************************************************************/
    IF (SELECT object_id('dbo.Get_Email_SP')) is not null
    DROP PROCEDURE dbo.Get_Email_SP;
    GO

    CREATE PROCEDURE dbo.Get_Email_SP
        @CreatedEmailAddress VARCHAR(254),
        @TagName VARCHAR(12)
    AS
    BEGIN
        SELECT  Account.AccountId,
                EmailReceipt.AccountIdSend,
                Account.UserName,
                Account.CreatedEmailAddress,
                --Join on the sending account id
                (Select a.Avatar
                    FROM dbo.Account a
                    WHERE a.AccountId = EmailReceipt.AccountIdSend) as AvatarSend,
                (Select a.UserName
                    FROM dbo.Account a
                    WHERE a.AccountId = EmailReceipt.AccountIdSend) as UserNameSend,

                (Select a.CreatedEmailAddress
                    FROM dbo.Account a
                    WHERE a.AccountId = EmailReceipt.AccountIdSend) as
                      CreatedEmailAddressSend,
                Email.RecvEmailList,
                (SELECT COUNT(*) FROM STRING_SPLIT(RecvEmailList,';')) AS
                  RecvEmailCount,
```

```
                Email.EmailId,
                Email.EmailSubject,
                Email.EmailBody,
                Email.DateTimeStamp
        FROM dbo.Account
        JOIN dbo.EmailReceipt ON EmailReceipt.AccountIdRecv = Account.AccountId
        JOIN dbo.Email on Email.EmailId = EmailReceipt.EmailId
        JOIN dbo.Tags ON Tags.TagId = EmailReceipt.TagId
        WHERE Account.CreatedEmailAddress = @CreatedEmailAddress
        -- LIKE will allow selecting using a wildcard.
        --(ie. @TagName = '%' returns emails from for the Account in ALL folders)
        AND Tags.TagName LIKE @TagName
    END
    GO
/*************************************************************************
 *    Description: Procedure to SELECT sent emails
 *
 *    Parameters that must be passed in:
 *        @CreatedEmailAddress
 *        @TagName
 *
 *************************************************************************/
    IF (SELECT object_id('dbo.Get_Sent_Email_SP')) is not null
    DROP PROCEDURE dbo.Get_Sent_Email_SP;
    GO

    CREATE PROCEDURE dbo.Get_Sent_Email_SP
        @CreatedEmailAddress VARCHAR(254),
        @TagName VARCHAR(12)='Sent'
    AS
    BEGIN
        DECLARE @AccountId INT;
        SELECT @AccountId = Account.AccountId
        FROM dbo.Account
        WHERE Account.CreatedEmailAddress = @CreatedEmailAddress

        SELECT
                Account.AccountId,
                Account.UserName AS UserNameSend,
                Account.CreatedEmailAddress AS CreatedEmailAddressSend,
                Account.Avatar AS AvatarSend,
                Account.UserName,
                Account.CreatedEmailAddress,
                Email.EmailId,
                Email.RecvEmailList,
                (SELECT COUNT(*) FROM STRING_SPLIT(RecvEmailList,';')) AS
                  RecvEmailCount,
                Email.EmailSubject,
                Email.EmailBody,
                Email.DateTimeStamp
        FROM dbo.Email
        JOIN dbo.Account ON Account.AccountId = Email.AccountId
```

```
        WHERE Email.AccountId = @AccountId
    END
    GO
/*************************************************************************
 *    Description: Procedure to SELECT sent emails with
 *                 the given TagName.
 *
 *    Parameters that must be passed in:
 *        @TagName
 *
 *************************************************************************/
    IF (SELECT object_id('dbo.Get_Email_With_Tag_SP')) is not null
    DROP PROCEDURE dbo.Get_Email_With_Tag_SP;
    GO

    CREATE PROCEDURE dbo.Get_Email_With_Tag_SP
        @TagName VARCHAR(12)=''
    AS
    BEGIN
        SELECT  EmailReceipt.EmailReceiptId,
                Email.EmailId,
                Email.AccountId AS AccountIdSend,
                (SELECT a.CreatedEmailAddress
                    FROM dbo.account a
                    WHERE a.AccountId = Email.AccountId) as CreatedEmailAddressSend,
                EmailReceipt.AccountIdSend as AccountIdRecv,
                (SELECT b.CreatedEmailAddress
                    FROM dbo.account b
                    WHERE b.AccountId = EmailReceipt.AccountIdSend) as
                      CreatedEmailAddressRecv,
                Email.EmailSubject,
                Email.EmailBody,
                Email.DateTimeStamp
        FROM EmailReceipt
        JOIN dbo.Email on Email.EmailId = EmailReceipt.EmailId
        JOIN dbo.Tags ON Tags.TagId = EmailReceipt.TagId
        WHERE Tags.TagName = @TagName
    END
    GO
/*************************************************************************
 *    Description: Procedure to SELECT all emails for all Accounts
 *                 with the given TagName.
 *
 *    Parameters that must be passed in:
 *        @TagName
 *
 *************************************************************************/
    IF (SELECT object_id('dbo.Get_Accounts_With_Flagged_Email_SP')) is not null DROP
      PROCEDURE dbo.Get_Accounts_With_Flagged_Email_SP;
    GO

    CREATE PROCEDURE dbo.Get_Accounts_With_Flagged_Email_SP
```

```
    AS
    BEGIN
        SELECT  DISTINCT
                Account.AccountId,
                Account.UserName,
                Account.Avatar,
                Account.PhoneNumber,
                Account.UserAddress,
                Account.CreatedEmailAddress,
                Account.Active
        FROM EmailReceipt
        JOIN dbo.Email on Email.EmailId = EmailReceipt.EmailId
        JOIN dbo.Tags ON Tags.TagId = EmailReceipt.TagId
        JOIN dbo.Account on Account.AccountId = Email.AccountId
        WHERE Tags.TagName = 'Flag'
    END
    GO
/*************************************************************************
 *    Description: Procedure to create a user-defined tag.
 *
 *    Parameters that must be passed in:
 *        @CreatedEmailAddress
 *        @TagName
 *
 *************************************************************************/
    IF (SELECT object_id('dbo.Create_Tag_SP')) is not null
    DROP PROCEDURE dbo.Create_Tag_SP;
    GO

    CREATE PROCEDURE dbo.Create_Tag_SP
        @CreatedEmailAddress VARCHAR(254)='', --1 byte per char + 2 bytes to hold
          length
        @TagName VARCHAR(12)=''
    AS
    BEGIN TRANSACTION
        BEGIN
            DECLARE @AccountId INT
            SELECT @AccountId = AccountId
                FROM dbo.Account
                WHERE CreatedEmailAddress = @CreatedEmailAddress;
            DECLARE @Now DATETIME
            SET @Now = GETDATE()
            -- if the tag already exists rollback and return -1
            IF (EXISTS(SELECT *
                        FROM dbo.Tags
                        WHERE Tags.AccountId = @AccountId
                            AND Tags.TagName = @TagName))
                GOTO On_Create_Tag_Error

            INSERT INTO dbo.Tags
                    (Tags.TagName,
                    Tags.TagType,
```

```
                    Tags.AccountId,
                    Tags.DateTimeStamp)
            values (@TagName,
                    'Custom',
                    @AccountId,
                    @Now)
            IF @@ERROR = -1 GOTO On_Create_Tag_Error

            COMMIT TRANSACTION
        END
        RETURN 0
        On_Create_Tag_Error:
            ROLLBACK TRANSACTION
            RETURN -1
    GO
/*****************************************************************************
 *    Description: Procedure to SELECT user-defined custom tags.
 *
 *    Parameters that must be passed in:
 *        @CreatedEmailAddress
 *        @TagType
 *
 *****************************************************************************/
    IF (SELECT object_id('dbo.Get_Tags_SP')) is not null
    DROP PROCEDURE dbo.Get_Tags_SP;
    GO

    CREATE PROCEDURE dbo.Get_Tags_SP
        @CreatedEmailAddress VARCHAR(254)='', --1 byte/char + 2 for length
        @TagType VARCHAR(6)=''
    AS
        BEGIN
            DECLARE @AccountId INT,
                    @Model VARCHAR(6),
                    @Custom VARCHAR(6)

            SELECT @AccountId = AccountId
                FROM dbo.Account
                WHERE CreatedEmailAddress = @CreatedEmailAddress;

            -- 'All' will match TagTypes 'Model' and 'Custom'
            -- 'Model' will match TagTypes 'Model' but wont match 'NotCustom'
            -- 'Custom' wont match TagTypes 'NotModel' but will match 'Custom'
            IF @TagType = 'All'
                BEGIN
                    SET @Model = 'Model'
                    SET @Custom ='Custom'
                END
            IF @TagType = 'Model'
                BEGIN
                    SET @Model = 'Model'
                    SET @Custom ='NotCustom'
```

```
            END
        IF @TagType = 'Custom'
            BEGIN
            SET @Model = 'NotModel'
            SET @Custom ='Custom'
        END

        SELECT
            TagId,
            TagName
        FROM dbo.Tags
        WHERE Tags.AccountId = @AccountId
            AND Tags.TagType in (SELECT @Model UNION SELECT @Custom)
    END
GO
/*************************************************************************
 *    Description: Procedure to UPDATE EmailRecipt tag.
 *
 *    Parameters that must be passed in:
 *      @AccountId
 *      @EmailId
 *      @TagName
 *
 *************************************************************************/
    IF (SELECT object_id('dbo.EmailRecipt_Tag_Update_SP')) is not null
    DROP PROCEDURE dbo.EmailRecipt_Tag_Update_SP;
    GO

    CREATE PROCEDURE dbo.EmailRecipt_Tag_Update_SP
        @AccountId INT=0,
        @EmailId INT=0,
        @TagName VARCHAR(6)=''
    AS
    BEGIN TRANSACTION
        BEGIN
            DECLARE @TagIdNew INT
            SELECT @TagIdNew = TagId
                FROM dbo.Tags
                WHERE Tags.TagName = @TagName AND Tags.AccountId = @AccountId
            IF @@Error = -1 GOTO On_EmailRecipt_Tag_Update_SP_Error

            UPDATE dbo.EmailReceipt
                SET TagId = @TagIdNew
                FROM EmailReceipt
                JOIN Tags ON EmailReceipt.TagId = Tags.TagId
                WHERE EmailReceipt.AccountIdRecv = @AccountId
                AND EmailReceipt.EmailId = @EmailId
            IF @@Error = -1 GOTO On_EmailRecipt_Tag_Update_SP_Error

        COMMIT TRANSACTION
        END
        RETURN 0
```

```
        On_EmailRecipt_Tag_Update_SP_Error:
            ROLLBACK TRANSACTION
            RETURN -1
    GO
/*****************************************************************************
 *    Description: Procedure to UPDATE Account table Active field.
 *
 *    Parameters that must be passed in:
 *      @AccountId
 *      @Active
 *
 *****************************************************************************/
    IF (SELECT object_id('dbo.Account_Active_Update_SP')) is not null
    DROP PROCEDURE dbo.Account_Active_Update_SP;
    GO

    CREATE PROCEDURE dbo.Account_Active_Update_SP
        @AccountId INT=0,
        @Active VARCHAR(5)=''
    AS
    BEGIN TRANSACTION
        BEGIN

            UPDATE dbo.Account
            SET Account.Active = @Active
            WHERE Account.AccountId = @AccountId
            IF @@Error = -1 GOTO On_Account_Active_Update_SP_Error

        COMMIT TRANSACTION
        END
        RETURN 0
        On_Account_Active_Update_SP_Error:
            ROLLBACK TRANSACTION
            RETURN -1
    GO
/*****************************************************************************
 *  Insert starter Account data
 *****************************************************************************/
    dbo.Account_Insert_SP
        @UserName=
            'James S',
        @UserAddress=
            '101 Main, Philadelphia, PA 01234',
        @PhoneNumber=
            '+11234567890',
        @CreatedEmailAddress=
            'jims@temple.edu',
        @ContactEmailAddress=
            'tun49199@temple.edu',
        @Avatar=
            3,
        --lower case 'p'
```

```
        @AccountPassword=
            0x2673BA5EA47ADBACDC45E9D9B2EF6B2B,
        @Active=
            'yes',
        @DateTimeStamp=
            NULL,
        @AccountRoleType=
            'User';
        GO
    dbo.Account_Insert_SP
        @UserName=
            'James S (admin)',
        @UserAddress=
            '101 Main, Philadelphia, PA 01234',
        @PhoneNumber=
            '+11234567890',
        @CreatedEmailAddress=
            'jims-admin@temple.edu',
        @ContactEmailAddress=
            'jims@gmail.com',
        @Avatar=
            3,
        @AccountPassword=
            0x2673BA5EA47ADBACDC45E9D9B2EF6B2B,
        @Active=
            'yes',
        @DateTimeStamp=
            NULL,
        @AccountRoleType=
            'Administrator';
        GO
    dbo.Account_Insert_SP
        @UserName=
            'Richard G',
        @UserAddress=
            '1712 Broad St, Philadelphia, PA 01234',
        @PhoneNumber=
            '123-312-0312',
        @CreatedEmailAddress=
            'richardg@temple.edu',
        @ContactEmailAddress=
            'richardg@gmail.com',
        @Avatar=
            4,
        @AccountPassword=
            0x2673BA5EA47ADBACDC45E9D9B2EF6B2B,
        @Active=
            'yes',
        @DateTimeStamp=
            NULL,
        @AccountRoleType=
            'User';
```

```
        GO
    dbo.Account_Insert_SP
        @UserName=
            'Prof P',
        @UserAddress=
            '1400 Pattison Ave, Philadelphia, PA 01234',
        @PhoneNumber=
            '123-858-5225',
        @CreatedEmailAddress=
            'prof@temple.edu',
        @ContactEmailAddress=
            'prof@yahoo.com',
        @Avatar=
            8,
        @AccountPassword=
            0x2673BA5EA47ADBACDC45E9D9B2EF6B2B,
        @Active=
            'yes',
        @DateTimeStamp=
            NULL,
        @AccountRoleType=
            'User';
        GO
    dbo.Account_Insert_SP
        @UserName=
            'Bruce W',
        @UserAddress=
            '1234B 1/2 E Independence Mall S Ste 12A',
        @PhoneNumber=
            '123-359-7563',
        @CreatedEmailAddress=
            'brucew@temple.edu',
        @ContactEmailAddress=
            'brucew@outlook.com',
        @Avatar=
            11,
        @AccountPassword=
            0x2673BA5EA47ADBACDC45E9D9B2EF6B2B,
        @Active=
            'yes',
        @DateTimeStamp=
            NULL,
        @AccountRoleType=
            'User';
        GO
    dbo.Account_Insert_SP
        @UserName=
            'Bruce W (admin)',
        @UserAddress=
            '1234B 1/2 E Independence Mall S Ste 12A',
        @PhoneNumber=
            '123-359-7563',
```

```
        @CreatedEmailAddress=
            'brucew-admin@temple.edu',
        @ContactEmailAddress=
            'brucew@outlook.com',
        @Avatar=
            11,
        @AccountPassword=
            0x2673BA5EA47ADBACDC45E9D9B2EF6B2B,
        @Active=
            'yes',
        @DateTimeStamp=
            NULL,
        @AccountRoleType=
            'Administrator';
        GO
--/*************************************************************************
-- *    Send/Create some sample emails
-- *************************************************************************/
    DECLARE @GetDate DATETIME
    SET @GetDate = GETDATE();

    exec dbo.Email_Send_SP
    @SendEmailAddress = 'prof@temple.edu',
    @RecvEmailList = 'jims@temple.edu',
    @EmailSubject = 'database',
    @EmailBody = 'There, it should be working, again',
    @DateTimeStamp = @GetDate

    exec dbo.Email_Send_SP
    @SendEmailAddress = 'brucew@temple.edu',
    @RecvEmailList = 'jims@temple.edu;richardg@temple.edu',
    @EmailSubject = 'Lecture Wednesday',
    @EmailBody = 'Discuss .NET Core WebAPIs',
    @DateTimeStamp = @GetDate

    exec dbo.Email_Send_SP
    @SendEmailAddress = 'richardg@temple.edu',
    @RecvEmailList = 'jims@temple.edu',
    @EmailSubject='Lorem ipsum',
    @EmailBody='Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
      eiusmod tempor.',
    @DateTimeStamp = @GetDate

    exec dbo.Email_Send_SP
    @SendEmailAddress='jims@temple.edu',
    @RecvEmailList = 'brucew@temple.edu;richardg@temple.edu',
    @EmailSubject='pellentesque',
    @EmailBody='At tellus at urna condimentum mattis pellentesque id. Sed adipiscing
      diam donec adipiscing.',
    @DateTimeStamp = @GetDate

    exec dbo.Email_Send_SP
```

```
    @SendEmailAddress='jims@temple.edu',
    @RecvEmailList = 'brucew@temple.edu',
    @EmailSubject='commodo viverra',
    @EmailBody='Malesuada nunc vel risus commodo viverra. Habitasse platea dictumst
      vestibulum rhoncus.',
    @DateTimeStamp = @GetDate
```