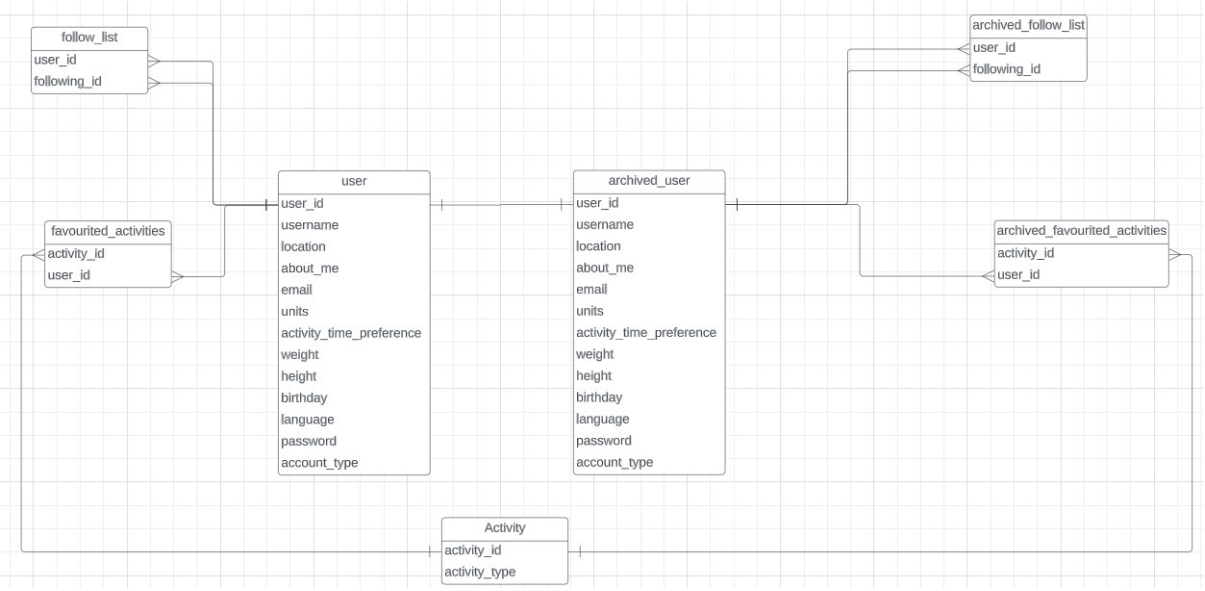
Comp 2001 Set Exercises Report

Entity Relationship Diagram



Assumptions made:

There is a one-to-one relationship between user and archived user, this will only happen for a short period of time when the user is getting archived and the data is stored in both tables.

Normalisation

UNF	
user_id	pk
username	
location	
about_me	
email	
units	
activity_time_preference	
weight	
height	
bithday	
language	
password	
account_type	
activity_id	
following_id	
activity_type	

1st normal form					
user_id	pk	activity_id	pk	activity_id	pk
username		user_id	pk	activity_type	
location					
about_me					
email					
units					
activity_time_preference					
weight					
height					
bithday					
language					
password					
account_type					

2nd normal form					
user_id	pk	activity_id	pk	activity_id	pk
username		user_id	pk	activity_type	
location					
about_me					
email					
units					
activity_time_preference					
weight					
height					
bithday					
language					
password					
account_type					

3rd normal form					
user_id		activity_id	pk	activity_id	pk
username		user_id	pk	activity_type	
location					
about_me					
email					
units					
activity_time_preference					
weight					
height					
bithday					
language					
password					
account_type					

SQL Commands for creating tables

```
CREATE SCHEMA CW1


CREATE TABLE CW1.[User] (
   user_id INTEGER PRIMARY KEY IDENTITY(1,1),
   username VARCHAR(50),
   email VARCHAR(80),
   password VARCHAR(40),
   account_type VARCHAR(15)
);
```

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
=	user_id	int	☑	☐		🗑	···
=	username	varchar(50)	☐	☑		🗑	···
=	email	varchar(80)	☐	☑		🗑	···
=	password	varchar(40)	☐	☑		🗑	···
=	account_type	varchar(15)	☐	☑		🗑	···

```
CREATE TABLE CW1.[Archive_User] (
   user_id INTEGER PRIMARY KEY,
   username VARCHAR(50),
   email VARCHAR(80),
   password VARCHAR(40),
   account_type VARCHAR(15)
);
```

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
=	user_id	int	☑	☐		🗑	···
=	username	varchar(50)	☐	☑		🗑	···
=	email	varchar(80)	☐	☑		🗑	···
=	password	varchar(40)	☐	☑		🗑	···
=	account_type	varchar(15)	☐	☑		🗑	···

```
CREATE TABLE CW1.[Activity] (
         activity_id INTEGER PRIMARY KEY IDENTITY(1,1),
         activity_type VARCHAR(30)
);
```

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
=	activity_id	int	☑	☐		🗑	···
=	activity_type	varchar(30)	☐	☑		🗑	···

```
CREATE TABLE CW1.[Favourite_Activities] (
   activity_id INTEGER,
   user_id INTEGER,
         FOREIGN KEY (activity_id) REFERENCES CW1.[Activity](activity_id),
         FOREIGN KEY (user_id) REFERENCES CW1.[User](user_id)
);
```

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
=	activity_id	int	☐	☑		🗑	···
=	user_id	int	☐	☑		🗑	···

```
CREATE TABLE CW1.[Archive_Favourite_Activities] (
   activity_id INTEGER,
   user_id INTEGER,
         FOREIGN KEY (activity_id) REFERENCES CW1.[Activity](activity_id),
         FOREIGN KEY (user_id) REFERENCES CW1.[Archive_User](user_id)
);
```

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
=	activity_id	int	☐	☑		🗑	···
=	user_id	int	☐	☑		🗑	···

```
CREATE TABLE CW1.[Follow_List] (
        user_id INTEGER,
        FOREIGN KEY (user_id) REFERENCES CW1.[User](user_id),
        follow_id INTEGER,
        FOREIGN KEY (follow_id) REFERENCES CW1.[User](user_id),
);
```

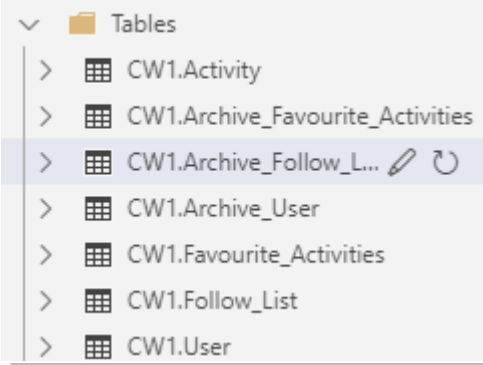
Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
=	user_id	int	☐	☑		🗑	···
=	follow_id	int	☐	☑		🗑	···

```
CREATE TABLE CW1.[Archive_Follow_List] (
        user_id INTEGER,
        follow_id INTEGER,
);
```

The reason that this doesn't not have any foreign keys to the Archive_User is that if there were, referential integrity would be broken and therefore I would not be able to implement this.

Move	Name	Type	Primary Key	Allow Nulls	Default Value	Remove	More Actions
=	user_id	int	☐	☑		🗑	···
=	follow_id	int	☐	☑		🗑	···

Result:



Activity	CW1	Table	···
Archive_Favourite_Activities	CW1	Table	···
Archive_Follow_List	CW1	Table	···
Archive_User	CW1	Table	···
Favourite_Activities	CW1	Table	···
Follow_List	CW1	Table	···
User	CW1	Table	···

Creating Views

```
CREATE VIEW CW1.[Main_View] AS

SELECT u.username AS "Username", u.email AS "Email", u.account_type AS "Account Type",

(SELECT COUNT(*) FROM CW1.[Follow_List] f WHERE u.user_id = f.user_id) Following,

(SELECT COUNT(*) FROM CW1.[Follow_List] f WHERE u.user_id = f.follow_id) Followers,

STRING_AGG(a.activity_type,', ') AS "Favourite Activities"

FROM CW1.[User] u

LEFT JOIN CW1.[Favourite_Activities] f ON u.user_id = f.user_id

LEFT JOIN CW1.[Activity] a ON f.activity_id = a.activity_id

GROUP BY u.user_id, u.account_type, u.username, u.email


SELECT * FROM CW1.[Main_View]




CREATE VIEW CW1.[Archive_View] AS

SELECT u.username AS "Archive Username", u.email AS "Archive Email", u.account_type AS
"Archive Account Type",

(SELECT COUNT(*) FROM CW1.[Archive_Follow_List] f WHERE u.user_id = f.user_id) Following,

(SELECT COUNT(*) FROM CW1.[Archive_Follow_List] f WHERE u.user_id = f.follow_id) Followers,

STRING_AGG(a.activity_type,', ') AS "Favourite Activities"

FROM CW1.[Archive_User] u

LEFT JOIN CW1.[Archive_Favourite_Activities] f ON u.user_id = f.user_id

LEFT JOIN CW1.[Activity] a ON f.activity_id = a.activity_id

GROUP BY u.user_id, u.account_type, u.username, u.email


SELECT * FROM CW1.[Archive_View]
```

Result:

Results Messages

	Username	Email	Account Type	Following	Followers	Favourite Activ…

	Archive Username	Archive Email	Archive Account…	Following	Followers	Favourite Activ…

With Data inputted:

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	1	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	0	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming

	Archive Username	Archive Email	Archive Account Type	Following	Followers	Favourite Activities
1	George	George@students.plymouth.ac.uk	admin	1	4	Running, Walking, Hiking

Commands to insert data:

```
INSERT INTO CW1.[User] ([username],[email],[password],[account_type])

VALUES

    ('Grace Hopper', 'grace@plymouth.ac.uk', 'ISAD123!', 'user'),

    ('Tim Berners-Lee', 'tim@plymouth.ac.uk', 'COMP2001!', 'user'),

    ('Veraint', 'Veraint@plymouth.ac.uk', 'COMP2001!', 'admin'),

    ('Patryk', 'pat@plymouth.ac.uk', 'patpassword!', 'admin'),

    ('Ada Lovelace', '@plymouth.ac.uk', 'insecurePassword', 'user'),

    ('George', 'George@students.plymouth.ac.uk', 'Georges password', 'admin');


EXEC CW1.[Follow_User] 1,2

EXEC CW1.[Follow_User] 4,5

EXEC CW1.[Follow_User] 5,4

EXEC CW1.[Follow_User] 2,1

EXEC CW1.[Follow_User] 3,1

EXEC CW1.[Follow_User] 1,6

EXEC CW1.[Follow_User] 2,6
```

```
EXEC CW1.[Follow_User] 3,6
EXEC CW1.[Follow_User] 4,6
EXEC CW1.[Follow_User] 6,3

EXEC CW1.[Favourite_Activity] 1,1
EXEC CW1.[Favourite_Activity] 1,2
EXEC CW1.[Favourite_Activity] 2,2
EXEC CW1.[Favourite_Activity] 2,3
EXEC CW1.[Favourite_Activity] 3,4
EXEC CW1.[Favourite_Activity] 3,1
EXEC CW1.[Favourite_Activity] 4,1
EXEC CW1.[Favourite_Activity] 4,5
EXEC CW1.[Favourite_Activity] 1,6
EXEC CW1.[Favourite_Activity] 2,6
EXEC CW1.[Favourite_Activity] 5,6
```

Stored procedures

Add User

This procedure is used to add a user to the Users table

```
CREATE PROCEDURE CW1.[Add_User]
@username VARCHAR(50),
@email VARCHAR(80),
@password VARCHAR(40),
@account_type VARCHAR(15)
AS
BEGIN
INSERT INTO CW1.[User] (username,email,password,account_type)
VALUES
```

```
    (@username, @email, @password, @account_type);
END
```

Code used to execute command:

```
EXEC CW1.[Add_User] 'Test_User', 'Test@students.plymouth.ac.uk', 'testpassword', 'user'
```

Result:

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	0	0	NULL
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	0	0	NULL
3	Veraint	Veraint@plymouth.ac.uk	admin	0	0	NULL
4	Patryk	pat@plymouth.ac.uk	admin	0	0	NULL
5	Ada Lovelace	@plymouth.ac.uk	user	0	0	NULL
6	Test_User	Test@students.plymouth.ac.uk	user	0	0	NULL

As you can see the user has been added to the table with the correct data.

Edit Username

This procedure is used to edit a username of a user in the Users table

```
CREATE PROCEDURE CW1.[Edit_Username]
@user_id INTEGER,
@new_username VARCHAR(50)
AS
BEGIN
UPDATE CW1.[User]
SET username = @new_username
WHERE user_id = @user_id;
END
```

Code used to execute command:

```
EXEC CW1.[Edit_Username] 6, "George"
```

This should change the user number 6's (Test User) username to be "George"

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	0	0	NULL
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	0	0	NULL
3	Veraint	Veraint@plymouth.ac.uk	admin	0	0	NULL
4	Patryk	pat@plymouth.ac.uk	admin	0	0	NULL
5	Ada Lovelace	@plymouth.ac.uk	user	0	0	NULL
6	George	Test@students.plymouth.ac.uk	user	0	0	NULL

As you can see this has worked correctly.

Edit Email

This procedure is used to edit the email of a user in the Users table

```
CREATE PROCEDURE CW1.[Edit_Email]

@user_id INTEGER,

@new_email VARCHAR(80)

AS

BEGIN

UPDATE CW1.[User]

SET email = @new_email

WHERE user_id = @user_id;

END
```

Code used to execute command:

```
EXEC CW1.[Edit_Email] 6, "George@students.plymouth.ac.uk"
```

This should change the user number 6's (George) email to be "George@students.plymouth.ac.uk"

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	0	0	NULL
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	0	0	NULL
3	Veraint	Veraint@plymouth.ac.uk	admin	0	0	NULL
4	Patryk	pat@plymouth.ac.uk	admin	0	0	NULL
5	Ada Lovelace	@plymouth.ac.uk	user	0	0	NULL
6	George	George@students.plymouth.ac.uk	user	0	0	NULL

As you can see George's email address is now set to what we wanted

Edit Password

This procedure is used to edit a user's password in the Users table

```
CREATE PROCEDURE CW1.[Edit_Password]
```

```
@user_id INTEGER,
@new_password VARCHAR(40)
AS
BEGIN
UPDATE CW1.[User]
SET password = @new_password
WHERE user_id = @user_id;
END
```

Code used to execute command:

```
EXEC CW1.[Edit_Password] 6, "Georges password"
```

This should change the user number 6's (George) password to be "Georges password".

	user_id	username	email	password	account_type
1	1	Grace Hopper	grace@plymouth.ac.uk	ISAD123!	user
2	2	Tim Berners-Lee	tim@plymouth.ac.uk	COMP2001!	user
3	3	Veraint	Veraint@plymouth.ac.uk	COMP2001!	admin
4	4	Patryk	pat@plymouth.ac.uk	patpassword!	admin
5	5	Ada Lovelace	@plymouth.ac.uk	insecurePassword	user
6	6	George	George@students.plymouth.ac.uk	Georges password	user

For this I had to use a different view instead of the views I created, here I am doing SELECT * FROM CW1.[User]. This is because the views that I created do not show the users password. However the data is still correct despite this and George's password has been changed to what we wanted.

Edit Account Type

This procedure will edit the account type of a user in the Users table.

```
CREATE PROCEDURE CW1.[Edit_Account_Type]
@user_id INTEGER,
@new_account_type VARCHAR(15)
AS
BEGIN
UPDATE CW1.[User]
```

```
SET account_type = @new_account_type

WHERE user_id = @user_id;

END
```

Code used to execute command:

```
EXEC CW1.[Edit_Account_Type] 6, "admin"
```

This should change the user number 6's (George) account type to be admin.

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	0	0	NULL
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	0	0	NULL
3	Veraint	Veraint@plymouth.ac.uk	admin	0	0	NULL
4	Patryk	pat@plymouth.ac.uk	admin	0	0	NULL
5	Ada Lovelace	@plymouth.ac.uk	user	0	0	NULL
6	George	George@students.plymouth.ac.uk	admin	0	0	NULL

As you can see George's account type is now admin.

Archive User

This procedure will archive a user, to do this we need to move all data that is relevant to the user into the archive tables. This requires the moving to be done in specific ordered steps as to keep referential integrity. To do this we must copy the user data, then favourite activities, then the follow lists, then finally we delete from the normal tables in the order of favourite activities, follow lists and then finally the user.

```
CREATE PROCEDURE CW1.[Archive_User_Procedure]

@user_id INTEGER

AS

BEGIN

INSERT INTO CW1.[Archive_User] (user_id, username, email, password, account_type)

SELECT *

FROM CW1.[User]

WHERE user_id = @user_id;


INSERT INTO CW1.[Archive_Favourite_Activities] (activity_id, user_id)

SELECT *

FROM CW1.[Favourite_Activities]

WHERE user_id = @user_id;
```

```
INSERT INTO CW1.[Archive_Follow_List] (user_id, follow_id)
SELECT *
FROM CW1.[Follow_List]
WHERE user_id = @user_id;

INSERT INTO CW1.[Archive_Follow_List] (user_id, follow_id)
SELECT *
FROM CW1.[Follow_List]
WHERE follow_id = @user_id;

DELETE FROM CW1.[Favourite_Activities]
WHERE user_id = @user_id;

DELETE FROM CW1.[Follow_List]
WHERE user_id = @user_id;

DELETE FROM CW1.[Follow_List]
WHERE follow_id = @user_id;

DELETE FROM CW1.[User]
WHERE user_id = @user_id;
END
```

This code creates the procedure, using insert and delete statements, also taking an input from the user to specify what user is to be archived.

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	2	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	2	1	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	1	1	Swimming
6	George	George@students.plymouth.ac.uk	admin	1	4	Running, Walking, Hiking

I have added some followers and activities to show how those also get archived.

Archiving the user:

EXEC CW1.[Archive_User_Procedure] 6

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	1	2	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	1	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	1	0	Walking
4	Patryk	pat@plymouth.ac.uk	admin	1	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	1	1	Swimming

	Archive Username	Archive Email	Archive Account Type	Following	Followers	Favourite Activities
1	George	George@students.plymouth.ac.uk	admin	1	4	Running, Walking, Hiking

As you can see the user has now been archived, preserving the followers that they have and still showing the favourite activities that they have. This means that the archive procedure is now fully working.

Here I could have created a procedure to un-archive a user, however I felt that since there was no mention of this in the specification that this was not a necessary procedure to implement.

Follow User

```
CREATE PROCEDURE CW1.[Follow_User]

@user_id INTEGER,

@follow_id INTEGER

AS

BEGIN


INSERT INTO CW1.[Follow_List] (user_id, follow_id)

VALUES

    (@user_id, @follow_id);

END
```

Before following:

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	1	2	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	1	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	1	0	Walking
4	Patryk	pat@plymouth.ac.uk	admin	1	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	1	1	Swimming

Code to follow the user:

EXEC CW1.[Follow_User] 5 , 3

EXEC CW1.[Follow_User] 4 , 3

EXEC CW1.[Follow_User] 2 , 3

EXEC CW1.[Follow_User] 1 , 3

This should mean that the user “Veraint” now has 4 followers, being all the other users in the table.

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	2	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	1	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming

As you can see Veraint now has 4 followers and all the other users in the table have increased their following count by 1.

Un-Follow User

```
CREATE PROCEDURE CW1.[Unfollow_User]

@user_id INTEGER,

@follow_id INTEGER

AS

BEGIN


DELETE FROM CW1.[Follow_List]

WHERE @user_id = user_id AND @follow_id = follow_id


END
```

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	2	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	1	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming

Currently user veraint is following user grace, we shall make veraint unfollow her.

5	3	1

This shows that user veriant (3) is following grace(1) in the followers table

Code to execute:

EXEC CW1.[Unfollow_User] 3 , 1

1	Grace Hopper	grace@plymouth.ac.uk	user	2	1	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	0	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming

Now veraints following count has gone down by 1, and graces followers count has also gone down by 1

Add Activity

```
CREATE PROCEDURE CW1.[Add_Activity]

@activity_type VARCHAR(30)

AS

BEGIN


INSERT INTO CW1.[Activity] (activity_type)

VALUES

    (@activity_type);

END


EXEC CW1.[Add_Activity] "Running"

EXEC CW1.[Add_Activity] "Walking"

EXEC CW1.[Add_Activity] "Cycling"

EXEC CW1.[Add_Activity] "Swimming"

EXEC CW1.[Add_Activity] "Hiking"
```

	activity_id	activity_type
1	1	Running
2	2	Walking
3	3	Cycling
4	4	Swimming
5	5	Hiking

Here all of the activities have been added to the Activity table

Favourite Activity

```
CREATE PROCEDURE CW1.[Favourite_Activity]
```

```
@user_id INTEGER,

@activity_id INTEGER

AS

BEGIN


INSERT INTO CW1.[Favourite_Activities] (activity_id, user_id)

VALUES

    (@user_id, @activity_id);

END
```

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	1	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	0	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming

Currently Ada Lovelace only has swimming as a favourite activity, we shall add running to that list.

```
EXEC CW1.[Favourite_Activity] 1, 5
```

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	1	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	0	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming, Running

Un-Favourite Activity

```
CREATE PROCEDURE CW1.[Un_Favourite_Activity]

@user_id INTEGER,

@activity_id INTEGER

AS

BEGIN


DELETE FROM CW1.[Favourite_Activities]

WHERE @user_id = user_id AND @activity_id = activity_id
```

END

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	1	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	0	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming, Running

Ada actually doesn't like running, so let's remove it:

EXEC CW1.[Un_Favourite_Activity] 5, 1

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	1	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	0	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming

Running has now been removed from Ada's list of favourite activities.

Triggers

```
CREATE TRIGGER CW1.[Email_Trigger]
ON CW1.[User]
AFTER INSERT
AS
BEGIN
   SET NOCOUNT ON;

   IF EXISTS (SELECT 1 FROM INSERTED WHERE CHARINDEX('@', email) = 0)
   BEGIN
      RAISERROR('Email is invalid', 16, 1);
      ROLLBACK;
   END
END;
```

To check that the trigger works correctly, we will add a user with an incorrect email address to test.
This is the code to add a user with an incorrect email address:

EXEC CW1.[Add_User] 'Test_User', 'Incorrect Email Address', 'testpassword', 'user'

```
1    EXEC CW1.[Add_User] 'Test_User', 'Incorrect Email Address', 'testpassword', 'user'
```

Results **Messages**

```
5:32:07 PM    Started executing query at Line 1
              (1 row affected)
              Msg 50000, Level 16, State 1, Procedure Email_Trigger, Line 9
              Email is invalid
              Msg 3609, Level 16, State 1, Procedure CW1.Add_User, Line 8
              The transaction ended in the trigger. The batch has been aborted.
              Total execution time: 00:00:00.058
```

As you can see the trigger does not let an email that is incorrect be added to the table, returning an error message saying email is invalid.

Here I will use a correct email format to show that it lets valid data in:

EXEC CW1.[Add_User] 'Test_User', 'Correct@email.com', 'testpassword', 'user'

	Username	Email	Account Type	Following	Followers	Favourite Activities
1	Grace Hopper	grace@plymouth.ac.uk	user	2	1	Running, Cycling, Swimming
2	Tim Berners-Lee	tim@plymouth.ac.uk	user	2	1	Running, Walking
3	Veraint	Veraint@plymouth.ac.uk	admin	0	4	Walking
4	Patryk	pat@plymouth.ac.uk	admin	2	1	Cycling
5	Ada Lovelace	@plymouth.ac.uk	user	2	1	Swimming
6	Test_User	Correct@email.com	user	0	0	NULL

Here you can see that the user has been added as they have a correct email address. There is no data for followers and activities because we need to add that later.

Appendix

```
CREATE SCHEMA CW1



CREATE TABLE CW1.[User] (
   user_id INTEGER PRIMARY KEY IDENTITY(1,1),
   username VARCHAR(50),
   email VARCHAR(80),
   password VARCHAR(40),
   account_type VARCHAR(15)
);



CREATE TABLE CW1.[Archive_User] (
   user_id INTEGER PRIMARY KEY,
   username VARCHAR(50),
   email VARCHAR(80),
   password VARCHAR(40),
   account_type VARCHAR(15)
);



CREATE TABLE CW1.[Activity] (
         activity_id INTEGER PRIMARY KEY IDENTITY(1,1),
         activity_type VARCHAR(30)
);



CREATE TABLE CW1.[Favourite_Activities] (
   activity_id INTEGER,
```

```
    user_id INTEGER,
        FOREIGN KEY (activity_id) REFERENCES CW1.[Activity](activity_id),
        FOREIGN KEY (user_id) REFERENCES CW1.[User](user_id)
);


CREATE TABLE CW1.[Archive_Favourite_Activities] (
    activity_id INTEGER,
    user_id INTEGER,
        FOREIGN KEY (activity_id) REFERENCES CW1.[Activity](activity_id),
        FOREIGN KEY (user_id) REFERENCES CW1.[Archive_User](user_id)
);


CREATE TABLE CW1.[Follow_List] (
        user_id INTEGER,
        FOREIGN KEY (user_id) REFERENCES CW1.[User](user_id),
        follow_id INTEGER,
        FOREIGN KEY (follow_id) REFERENCES CW1.[User](user_id),
);


CREATE TABLE CW1.[Archive_Follow_List] (
        user_id INTEGER,
        follow_id INTEGER,
);
```

SQL View

```
CREATE VIEW CW1.[Main_View] AS
SELECT u.username AS "Username", u.email AS "Email", u.account_type AS "Account Type",
```

```
(SELECT COUNT(*) FROM CW1.[Follow_List] f WHERE u.user_id = f.user_id) Following,

(SELECT COUNT(*) FROM CW1.[Follow_List] f WHERE u.user_id = f.follow_id) Followers,

STRING_AGG(a.activity_type,', ') AS "Favourite Activities"

FROM CW1.[User] u

LEFT JOIN CW1.[Favourite_Activities] f ON u.user_id = f.user_id

LEFT JOIN CW1.[Activity] a ON f.activity_id = a.activity_id

GROUP BY u.user_id, u.account_type, u.username, u.email


SELECT * FROM CW1.[Main_View]




CREATE VIEW CW1.[Archive_View] AS

SELECT u.username AS "Archive Username", u.email AS "Archive Email", u.account_type AS "Archive
Account Type",

(SELECT COUNT(*) FROM CW1.[Archive_Follow_List] f WHERE u.user_id = f.user_id) Following,

(SELECT COUNT(*) FROM CW1.[Archive_Follow_List] f WHERE u.user_id = f.follow_id) Followers,

STRING_AGG(a.activity_type,', ') AS "Favourite Activities"

FROM CW1.[Archive_User] u

LEFT JOIN CW1.[Archive_Favourite_Activities] f ON u.user_id = f.user_id

LEFT JOIN CW1.[Activity] a ON f.activity_id = a.activity_id

GROUP BY u.user_id, u.account_type, u.username, u.email


SELECT * FROM CW1.[Archive_View]

CREATE PROCEDURE CW1.[Add_User]

@username VARCHAR(50),

@email VARCHAR(80),

@password VARCHAR(40),

@account_type VARCHAR(15)

AS

BEGIN
```

```
INSERT INTO CW1.[User] (username,email,password,account_type)
VALUES
        (@username, @email, @password, @account_type);
END


EXEC CW1.[Add_User] 'Veraint', 'Veraint@students.plymouth.ac.uk', 'testpassword', 'admin'




CREATE PROCEDURE CW1.[Edit_Username]
@user_id INTEGER,
@new_username VARCHAR(50)
AS
BEGIN
UPDATE CW1.[User]
SET username = @new_username
WHERE user_id = @user_id;
END

EXEC CW1.[Edit_Username] 4, "Pat102"


CREATE PROCEDURE CW1.[Edit_Email]
@user_id INTEGER,
@new_email VARCHAR(80)
AS
BEGIN
UPDATE CW1.[User]
SET email = @new_email
```

```
WHERE user_id = @user_id;
END

EXEC CW1.[Edit_Email] 4, "Pat102@gmail.com"


CREATE PROCEDURE CW1.[Edit_Password]
@user_id INTEGER,
@new_password VARCHAR(40)
AS
BEGIN
UPDATE CW1.[User]
SET password = @new_password
WHERE user_id = @user_id;
END

EXEC CW1.[Edit_Password] 4, "Pats password"


CREATE PROCEDURE CW1.[Edit_Account_Type]
@user_id INTEGER,
@new_account_type VARCHAR(15)
AS
BEGIN
UPDATE CW1.[User]
SET account_type = @new_account_type
WHERE user_id = @user_id;
END

EXEC CW1.[Edit_Account_Type] 4, "user"
```

```
CREATE PROCEDURE CW1.[Archive_User_Procedure]
@user_id INTEGER
AS
BEGIN
INSERT INTO CW1.[Archive_User] (user_id, username, email, password, account_type)
SELECT *
FROM CW1.[User]
WHERE user_id = @user_id;

INSERT INTO CW1.[Archive_Favourite_Activities] (activity_id, user_id)
SELECT *
FROM CW1.[Favourite_Activities]
WHERE user_id = @user_id;

INSERT INTO CW1.[Archive_Follow_List] (user_id, follow_id)
SELECT *
FROM CW1.[Follow_List]
WHERE user_id = @user_id;

INSERT INTO CW1.[Archive_Follow_List] (user_id, follow_id)
SELECT *
FROM CW1.[Follow_List]
WHERE follow_id = @user_id;

DELETE FROM CW1.[Favourite_Activities]
WHERE user_id = @user_id;
```

```
DELETE FROM CW1.[Follow_List]
WHERE user_id = @user_id;

DELETE FROM CW1.[Follow_List]
WHERE follow_id = @user_id;

DELETE FROM CW1.[User]
WHERE user_id = @user_id;
END
```

```
EXEC CW1.[Archive_User_Procedure] 2
```

```
CREATE PROCEDURE CW1.[Follow_User]
@user_id INTEGER,
@follow_id INTEGER
AS
BEGIN

INSERT INTO CW1.[Follow_List] (user_id, follow_id)
VALUES
	(@user_id, @follow_id);
END
```

```
EXEC CW1.[Follow_User] 1 , 2
```

```
CREATE PROCEDURE CW1.[Unfollow_User]
@user_id INTEGER,
@follow_id INTEGER
AS
BEGIN

DELETE FROM CW1.[Follow_List]
WHERE @user_id = user_id AND @follow_id = follow_id

END

EXEC CW1.[Unfollow_User] 1 , 2
```

```
CREATE PROCEDURE CW1.[Add_Activity]
@activity_type VARCHAR(30)
AS
BEGIN

INSERT INTO CW1.[Activity] (activity_type)
VALUES
        (@activity_type);
END
```

```
EXEC CW1.[Add_Activity] "Running"
```

```
CREATE PROCEDURE CW1.[Favourite_Activity]
@user_id INTEGER,
@activity_id INTEGER
AS
BEGIN

INSERT INTO CW1.[Favourite_Activities] (activity_id, user_id)
VALUES
        (@user_id, @activity_id);
END

EXEC CW1.[Favourite_Activity] 3, 4
```

```
CREATE PROCEDURE CW1.[Un_Favourite_Activity]
@user_id INTEGER,
@activity_id INTEGER
AS
BEGIN

DELETE FROM CW1.[Favourite_Activities]
WHERE @user_id = user_id AND @activity_id = activity_id

END
```

```
EXEC CW1.[Un_Favourite_Activity] 3, 4


CREATE TRIGGER CW1.[Email_Trigger]
ON CW1.[User]
AFTER INSERT
AS
BEGIN
   SET NOCOUNT ON;


   IF EXISTS (SELECT 1 FROM INSERTED WHERE CHARINDEX('@', email) = 0)
   BEGIN
      RAISERROR('Email is invalid', 16, 1);
      ROLLBACK;
   END
END;
```