CPSC 304 Project Cover Page

Milestone #: ____2____

Date: _Mar.1st 2024___________

Group Number: ____61______

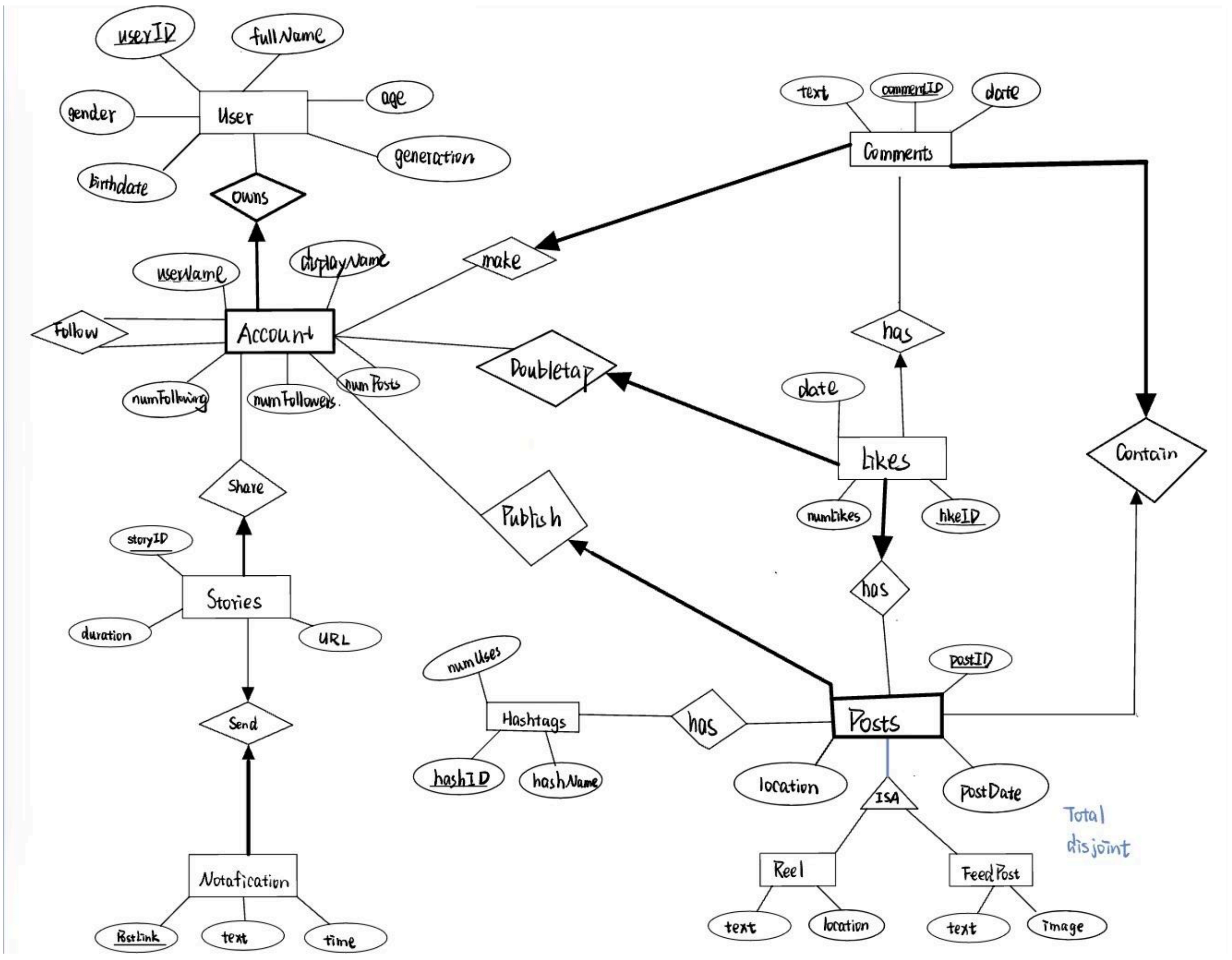
Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Zoey Ma	57920241	c6k9p	Ziyunma949@gmail.com
Edmond Ye	32019416	u8j0n	yegefei0121@gmail.com
Anna Tao	76542653	n5b4q	Annatao2004@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

2.
Our project is a database that will model social media users, accounts, the posts accounts can made and interactions made between these accounts. Our database models each individual user as an entity and link them with multiple accounts, we wish to store their profile information such as usernames, follower counts, following counts, as user-specific attributes.

Updated ER diagram:



3. Note for changes made to the ER diagram:

1. We broke down the Make relationship into several binary relationships in an attempt to make the domain clearer.
    a. We added a post relationship between Account and Posts that is 1:M, with total participation, meaning an account can post multiple posts, and every post must be associated with an account.
    b. We added a binary 1: M, Make relationship between Account and Comments, meaning an account can make many Comments, each comment must be associated with an account.
    c. We added a 1:1 Doubletap relationship between Account and Likes, with a total participation constraint on Likes, to indicate that every like must be performed by an Account.
    d. We added a Posts Has Likes relationship that is 1:M and total participation on Likes, meaning that every Like must be associated to a Post, and a Post can have multiple Likes.
    e. Instagram also has a feature where comments can have likes, So we added a Comments Has Likes Relationship which is 1:M, so Comments can have 0 or more likes.
        i. On a higher level we can see that every Like must be associated to one Account and associated to one Post. A Post can have many likes.
    f. Instagram allows you to comment on posts, so we added a Posts Contain Comments, that is 1:1 with total participation from Comments, so every Comment must be uniquely associated to some post.
        i. On a higher level we can see that every Comment must be associated to an Account and must be associated to a post
2. Changes the ISA relationship
    a. In instagram there are two types of posts you can make, therefore our designed captures:
        i. A Reel, which is a video-type of post and shows up in a separate section from feed posts on an account
        ii. A Feedpost which can contain one or more photo/videos and a caption.
        iii. Both of these post types share common attributes and share the ability to interact with other entities similarly
        iv. We chose Total and Disjoint constraints since a Post must fall under one of these two subclasses, and these two subclasses are separate categorizations
3. Added attributes
    a. To User, we added birthdate and generation (Boomers, Millennials, Generation , etc.). We felt that these would be important attributes to add to a user's table

4. schemas
NotificationsSend(<u>PostLink: </u>VARCHAR[ ], text: VARCHAR[ ], time: time, **storyID:** NOT NULL)

User(<u>userID:</u> INTEGER, gender: VARCHAR[ ], fullName: VARCHAR[ ], age: INTEGER, birthdate: Date, generation: VARCHAR[]);

StoriesShare(<u>storyID</u>: VARCHAR[ ], duration: date NOT NULL, URL: VARCHAR[ ] UNIQUE, **userName:** VARCHAR[ ]**, userID:** INTEGER NOT NULL)

Reel(text: CHAR[ ], length: CHAR[ ], **postID:** INTEGER, postID: INTEGER, location: VARCHAR[ ], postDate: date)

FeedPost(text: CHAR[ ], image, **postID:** INTEGER, location: VARCHAR[ ], postDate: date)

Hashtags(hashID: INTEGER, hashName: VARCHAR[ ])

HashtagHasPost(**hashID:** CHAR[ ], **postID:** INTEGER)

LikesDoubleTapHas(likeID: CHAR[ ], numLikes: INTEGER, date: date, **postID:** INTEGER NOT NULL**, userName:** VARCHAR[ ]**, userID:**INTEGER**, commentID:** CHAR[ ])

PublishPosts(PostID, location, post_date, **userName, userID**)

MakeComments(text: CHAR[ ], commentID: CHAR[ ], date: date, **userName:** VARCHAR[ ] NOT NULL**, userID**:INTEGER NOT NULL) // we need to include userName, userID since both of them are our primary key

Contain(**commentID:** CHAR[ ]**, postID:** INTEGER)

AccountOwns(userName: VARCHAR[ ], displayName: VARCHAR[ ], numFollowing: INTEGER, numFollowers: INTEGER, numPosts: INTEGER, **userID:** INTEGER)

5. Function Dependencies

(1) NotificationsSend(PostLink: VARCHAR[ ], text: VARCHAR[ ], time: time, **storyID:** NOT NULL)
FD:
PostLink -> text, time, storyID
(time, storyID) -> PostLink

(2) User(userID: INTEGER, gender: VARCHAR[ ], fullName: VARCHAR[ ], age: INTEGER, birthdate: Date, generation: CHAR[]);
FD:
userID-> gender, fullName, age, birthdate
birthdate-> generation

(3) StoriesShare(storyID: CHAR[ ], duration: date, URL: VARCHAR[ ] UNIQUE, **userName:** VARCHAR[ ]**, userID:** INTEGER)
FD:
storyID -> duration, URL, userName, userID
URL -> duration, storyID, userName, userID
(userID, duration) -> (storyID, URL, name) //// not in BNF

(4) Reel(text: CHAR[ ], length: CHAR[ ], postID: INTEGER, location: VARCHAR[ ], postDate: date)
FD:
postID -> length, text, location, postDate

(length, text, location, postDate) => hashID

(5) FeedPost(text: CHAR[ ], numImage: INTEGER, postID: INTEGER, location: VARCHAR[ ], postDate: date)
FD:
postID->text, numImage, location, postDate
(text, numImage, location, postDate) => hashID

(6) Hashtags(hashID: INTEGER, hashName: VARCHAR[ ], numUses: INTEGER)
FD:
hashID -> hashName, numUses
hashName -> numUses

(7) HashtagHasPost(**hashID:** INTEGER**, postID:** INTEGER)
FD:
hashID -> postID
postID -> length, text, location, postDate

(8) LikesDoubletapHas(likeID: CHAR[ ], numLikes: INTEGER, date: date, **postID:** INTEGER NOT NULL**, userName:** VARCHAR[ ]**, userID:**INTEGER**, commentID:** CHAR[ ])
FD:
likeID -> numLikes, date, postID, userName, userID, commentID
(postID, userID) -> likeID
userID->userName
postID-> userID, userName, numLikes, commentID
date, postID -> numLikes

(9) MakeComments(text: CHAR[ ], commentID: CHAR[ ], date: date, **userName:** VARCHAR[ ] NOT NULL**, userID:** INTEGER NOT NULL)
FD:
commentID -> text, date, userName, userID
userID->userName
userID, text, date-> commentID

(10) Contain(**commentID:** CHAR[ ]**, postID:** INTEGER)
FD:
commentID-> postId
postID -> commentID

(11)AccountOwns(userName: VARCHAR[ ], displayName: VARCHAR[ ], numFollowing: INTEGER, numFollowers: INTEGER, numPosts: INTEGER, **userID:** CHAR[ ])
FD:
userName, userID -> displayName, numFollowing, numFollowers, numPosts
displayName, numFollowing, numFollowers, numPosts -> username, userID

(12) PublishPosts(PostID, location, postDate, **userName, userID**)
FD:

PostID -> location, postDate, userName, userID
userID->userName

6. Normalization

(1) Both PostLink and (time, storyID) are superkeys, so no decomposition is required
NotificationsSend(<u>PostLink: </u>VARCHAR[ ], text: VARCHAR[ ], time: time, **storyID:** NOT NULL)

(2) Birthdate is not superkey, decompose the table into two tables. One contains birthdate and generation, and the other one contains all attributes except generation.
BirthdateGen(<u>birthdate</u>: DATE, generation: CHAR[])
User(<u>userID:</u> INTEGER, gender: VARCHAR[ ], fullName: VARCHAR[ ], age: INTEGER, birthdate: Date)

(3) Already in BCNF.
StoriesShare<u>(storyID</u>: VARCHAR[ ], duration: date, URL: VARCHAR[ ] UNIQUE, **userName:** VARCHAR[ ]**, userID:** INTEGER)

(4) Reel
Already in BCNF
Reel(text: CHAR[ ], length: CHAR[ ], <u>postID: </u>INTEGER, location: VARCHAR[ ], postDate: date)

(5) FeedPost
Already in BCNF
FeedPost(text: CHAR[ ], numImage: INTEGER, <u>postID: </u>INTEGER, location: VARCHAR[ ], postDate: date)

(6) Hashtags
hashname is not superkey, so it is not in BCNF, decomposes into two tables. One contains hashName and numUses, and the other one contains hashID and hashName.
Hashtags(<u>hashID: </u>INTEGER, hashName: VARCHAR[ ])
HashName(<u>hashName</u>: VARCHAR[ ], numUses: INTEGER)

(7) HashtagHasPost
Already in BCNF.
HashtagHasPost(<u>**hashID:**</u> INTEGER<u>**, postID:**</u> INTEGER])

(8) LikesDoubletapHas
UserID is not superkey, so it is not in BCNF. PostID is not superkey either but it is part of a minimal key, so it is in 3NF. We only need to decompose the FD userID->userName.
UserIdentity(<u>**userID**</u>**:**INTEGER,userName**:** VARCHAR[ ])
LikesDoubletapHas(<u>likeID: </u>CHAR[ ], numLikes: INTEGER, date: date, **postID:** INTEGER NOT NULL**,userID:**INTEGER**, commentID:** CHAR[ ])

(9) MakeComments

UserID is not superkey, but it is being decomposed in the above relationship, still decompose with this FD but get rid of the duplicated (UserID,UserName) table.
MakeComments(text: CHAR[ ], commentID: CHAR[ ], date: date**, userID:** INTEGER NOT NULL)

(10) Contain
Already in BCNF.
Contain(**commentID:** CHAR[ ]**, postID:** INTEGER)

(11) AccountOwns
Already in BCNF.
AccountOwns(userName: VARCHAR[ ], displayName: VARCHAR[ ], numFollowing: INTEGER, numFollowers: INTEGER, numPosts: INTEGER, **userID:** INTEGER)

(12) PublishPosts
UserID is not superkey, but it is being decomposed in the above relationship, still decompose with this FD but get rid of the duplicated (UserID,UserName) table.
PublishPosts(PostID: INTEGER, location: VARCHAR[ ], postDate: DATE, **userID**: INTEGER)

7. SQL DDL statements

(1)
```
CREATE TABLE NotificationsSend (
PostLink VARCHAR[255],
text VARCHAR[500],
time TIME,
storyID VARCHAR[255] NOT NULL,
PRIMARY KEY (PostLink),
FOREIGN KEY (storyID) REFERENCES StoriesShare,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

(2)
```
CREATE TABLE User(
userID INTEGER,
gender VARCHAR[3],
fullName VARCHAR[30],
age INTEGER,
birthdate DATE,
PRIMARY KEY (userID)
)

CREATE TABLE BirthdateGen(
birthdate DATE,
generation CHAR[4],
PRIMARY KEY (birthdate)
)
```

(3) CREATE TABLE StoriesShare(
storyID VARCHAR[255],
duration DATE,
URL VARCHAR[255] UNIQUE,
userName VARCHAR[30],
userID INTEGER,
PRIMARY KEY (storyID),
FOREIGN KEY (userName) REFERENCES AccountOwns,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (userID) REFERENCES UserIdentity,
ON UPDATE CASCADE
ON DELETE CASCADE
)

(4) CREATE TABLE Reel(
text CHAR[255],
length CHAR[255],
postID INTEGER,
location VARCHAR[255],
postDate DATE,
PRIMARY KEY (postID)
)

(5) CREATE TABLE FeedPost(
text CHAR[255],
numImage INTEGER,
postID INTEGER,
location VARCHAR[255],
postDate DATE,
PRIMARY KEY (postID)
)

(6) CREATE TABLE Hashtags(
hashID INTEGER,
hashName VARCHAR[100],
PRIMARY KEY (hashID)
)

CREATE TABLE HashName(
hashName: VARCHAR[100],
numUses: INTEGER,
PRIMARY KEY (hashName)
)

(7) CREATE TABLE HashtagHasPost(
hashID INTEGER,
postID INTEGER,

```
PRIMARY KEY (hashID,postID),
FOREIGN KEY (hashID) REFERENCES Hashtags,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (postID) REFERENCES FeedPost,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

(8)
```
CREATE TABLE UserIdentity(
userID INTEGER,
userName VARCHAR[100]
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User,
ON UPDATE CASCADE
ON DELETE CASCADE

CREATE TABLE LikesDoubletapHas(
likeID CHAR[11],
numLikes INTEGER,
date DATE,
postID INTEGER NOT NULL,
userID INTEGER,
commentID CHAR[11]
PRIMARY KEY (likeID),
FOREIGN KEY (postID) REFERENCES FeedPost,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (userID) REFERENCES UserIdentity,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (commentID) REFERENCES MakeComments,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

(9)
```
CREATE TABLE MakeComments(
text CHAR[100],
commentID CHAR[11],
date DATE,
userID INTEGER NOT NULL,
PRIMARY KEY (commentID),
FOREIGN KEY (userID) REFERENCES UserIdentity,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

```
(10)    CREATE TABLE Contain(
    commentID CHAR[11],
    postID INTEGER,
    PRIMARY KEY (commentID,postID),
    FOREIGN KEY (commentID) REFERENCES MakeComments,
    ON UPDATE CASCADE
    ON DELETE CASCADE
    FOREIGN KEY (postID) REFERENCES FeedPost,
    ON UPDATE CASCADE
    ON DELETE CASCADE
    )

(11) CREATE TABLE AccountOwns(
    userName VARCHAR[100],
    displayName VARCHAR[100],
    numFollowing INTEGER,
    numFollowers INTEGER,
    numPosts INTEGER,
    userID INTEGER,
    PRIMARY KEY (userName,userID),
    FOREIGN KEY (userID) REFERENCES UserIdentity,
    ON UPDATE CASCADE
    ON DELETE CASCADE
    )

(12)    CREATE TABLE PublishPosts(
    PostID INTEGER,
    location VARCHAR[100],
    postDate DATE,
    userID INTEGER,
    PRIMARY KEY (PostID),
    FOREIGN KEY (userID) REFERENCES UserIdentity,
    ON UPDATE CASCADE
    ON DELETE CASCADE
    )
```

8. Populated instances

TABLE NotificationsSend

```
INSERT
INTO NotificationsSend (PostLink, text, time, storyID)
VALUES ('https://test.com/post1', ‘this is post 1', '12:00:00', 'story1')
```

```
INSERT
INTO NotificationsSend (PostLink, text, time, storyID)
VALUES ('https://test.com/post2', 'this is post 2', '13:00:00', 'story2')

INSERT
INTO NotificationsSend (PostLink, text, time, storyID)
VALUES ('https://test.com/post3', this is post 3', '14:00:00', 'story3')

INSERT
INTO NotificationsSend (PostLink, text, time, storyID)
VALUES ('https://test.com/post4', this is post 4', '15:00:00', 'story4')

INSERT
INTO NotificationsSend (PostLink, text, time, storyID)
VALUES ('https://test.com/post5', 'this is post 5', '15:00:00', 'story5')
```

TABLE User

```
INSERT
INTO User (userID, gender, fullName, age, birthdate)
VALUES (1, 'F', 'Jane Doe', 28, '1995-04-12')

INSERT
INTO User (userID, gender, fullName, age, birthdate)
VALUES (2, 'M', 'John Smith', 32, '1991-08-24')

INSERT
INTO User (userID, gender, fullName, age, birthdate)
VALUES (3, 'O', 'Alex Johnson', 29, '1994-11-05')

INSERT
INTO User (userID, gender, fullName, age, birthdate)
VALUES (4, 'F', 'Emily Davis', 35, '1988-02-19')

INSERT
INTO User (userID, gender, fullName, age, birthdate)
VALUES (5, 'M', 'Chris Green', 40, '1983-07-30')
```

TABLE BirthdateGen

```
INSERT
INTO BirthdateGen (birthdate, generation)
VALUES ('1946-01-01', 'BB')

INSERT
INTO BirthdateGen (birthdate, generation)
VALUES ('1964-01-01', 'X')

INSERT
INTO BirthdateGen (birthdate, generation)
```

```
VALUES ('1980-01-01', 'M')

INSERT
INTO BirthdateGen (birthdate, generation)
VALUES ('1996-01-01', 'Z')

INSERT
INTO BirthdateGen (birthdate, generation)
VALUES ('2010-01-01', 'A')
```

TABLE StoriesShare

```
INSERT
INTO StoriesShare (storyID, duration, URL, userName, userID)
VALUES ('story1', '2024-03-01', 'http://test.com/story1', 'userOne', 12345678901)

INSERT
INTO StoriesShare (storyID, duration, URL, userName, userID)
VALUES ('story2', '2024-03-02', 'http://test.com/story2', 'userTwo', 12345678902)

INSERT
INTO StoriesShare (storyID, duration, URL, userName, userID)
VALUES ('story3', '2024-03-03', 'http://test.com/story3', 'userThree', 12345678903)

INSERT
INTO StoriesShare (storyID, duration, URL, userName, userID)
VALUES ('story4', '2024-03-04', 'http://test.com/story4', 'userFour', 12345678904)

INSERT
INTO StoriesShare (storyID, duration, URL, userName, userID)
VALUES ('story5', '2024-03-05', 'http://test.com/story5', 'userFive', 12345678905)
```

TABLE Reel

```
INSERT
INTO Reel (text, length, postID, location, postDate)
VALUES ('First reel’, '10', 10000001, 'New York', '2024-01-01')

INSERT
INTO Reel (text, length, postID, location, postDate)
VALUES (‘second  reel’, '10', 10000002, 'New York', '2024-01-02');
INSERT
INTO Reel (text, length, postID, location, postDate)
VALUES (‘second  reel’, '10', 10000003, 'New York', '2024-01-02')

INSERT
INTO Reel (text, length, postID, location, postDate)
VALUES (‘second  reel’, '10', 10000004, 'New York', '2024-01-02')

INSERT
```

INTO Reel (text, length, postID, location, postDate)
VALUES (‘second reel’, '10', 10000005, 'New York', '2024-01-02')

TABLE FeedPost
INSERT
INTO FeedPost (text, numImage, postID, location, postDate)
VALUES ('Post 1’, 2, 10000001, 'New York', '2024-03-01')

INSERT
INTO FeedPost (text, numImage, postID, location, postDate)
VALUES ('Post 2', 1, 10000002, 'Los Angeles', '2024-03-02')

INSERT
INTO FeedPost (text, numImage, postID, location, postDate)
VALUES ('Post 3', 3, 10000003, ‘New York, '2024-03-03')

INSERT
INTO FeedPost (text, numImage, postID, location, postDate)
VALUES ('Post 4', 4, 10000004, 'Miami', '2024-03-04')

INSERT
INTO FeedPost (text, numImage, postID, location, postDate)
VALUES ('Post 5', 5, 10000005, ‘New York, '2024-03-05')

TABLE Hashtags
INSERT
INTO Hashtags (hashID, hashName)
VALUES (1, '#travel')

INSERT
INTO Hashtags (hashID, hashName)
VALUES (2, '#food')

INSERT
INTO Hashtags (hashID, hashName)
VALUES (3, '#fitness')

INSERT
INTO Hashtags (hashID, hashName)
VALUES (4, '#technology')

INSERT
INTO Hashtags (hashID, hashName)
VALUES (5, '#fashion')

TABLE HashName
INSERT
INTO HashName (hashName, numUses)

```
VALUES ('#travel', 1200)

INSERT
 INTO HashName (hashName, numUses)
VALUES ('#food', 950)

INSERT
INTO HashName (hashName, numUses)
VALUES ('#fitness', 800)

INSERT
INTO HashName (hashName, numUses)
VALUES ('#technology', 620)

INSERT
INTO HashName (hashName, numUses)
VALUES ('#fashion', 890)

TABLE HashtagHasPost
INSERT
INTO HashtagHasPost (hashID, postID)
VALUES (1, 1)

INSERT
INTO HashtagHasPost (hashID, postID)
VALUES (2, 1)

INSERT
INTO HashtagHasPost (hashID, postID)
VALUES (3, 3)

INSERT
INTO HashtagHasPost (hashID, postID)
VALUES (4, 3)

INSERT
INTO HashtagHasPost (hashID, postID)
VALUES (5, 4)

TABLE UserIdentity
INSERT
INTO UserIdentity (userID, userName)
VALUES (1, 'UserOne')

INSERT
INTO UserIdentity (userID, userName)
VALUES (2, 'UserTwo')
```

```
INSERT
INTO UserIdentity (userID, userName)
VALUES (3, 'UserThree')

INSERT
INTO UserIdentity (userID, userName)
VALUES (4, 'UserFour')

INSERT
INTO UserIdentity (userID, userName)
VALUES (5, 'UserFive')
```

TABLE LikesDoubletapHas

```
INSERT
INTO LikesDoubletapHas (likeID, numLikes, date, postID, userID, commentID)
VALUES ('L000000001', 100, '2024-02-28', 1, 1, 'C00000001')

INSERT
INTO LikesDoubletapHas (likeID, numLikes, date, postID, userID, commentID)
VALUES ('L000000002', 150, '2024-02-29', 2, 2, 'C00000002')

INSERT
INTO LikesDoubletapHas (likeID, numLikes, date, postID, userID, commentID)
VALUES ('L000000003', 200, '2024-03-01', 3, 3, 'C00000003')

INSERT
INTO LikesDoubletapHas (likeID, numLikes, date, postID, userID, commentID)
VALUES ('L000000004', 250, '2024-03-02', 4, 4, 'C00000004')

INSERT
INTO LikesDoubletapHas (likeID, numLikes, date, postID, userID, commentID)
VALUES ('L000000005', 300, '2024-03-03', 5, 5, 'C00000005')
```

TABLE MakeComments

```
INSERT
INTO MakeComments (text, commentID, date, userID)
VALUES ('Great!', 'C000000001', '2024-03-01', 1);


INSERT
INTO MakeComments (text, commentID, date, userID)
VALUES ('Love this', 'C000000002', '2024-03-02', 2);

INSERT
INTO MakeComments (text, commentID, date, userID)
VALUES ('Amazing content', 'C000000003', '2024-03-03', 3);

INSERT
```

```
INTO MakeComments (text, commentID, date, userID)
VALUES ('So bad’', 'C000000004', '2024-03-04', 4);

INSERT
INTO MakeComments (text, commentID, date, userID)
VALUES ('work harder', 'C000000005', '2024-03-05', 5);
```

TABLE Contain

```
INSERT
INTO Contain (commentID, postID)
VALUES ('C000000001', 1);

INSERT
INTO Contain (commentID, postID)
VALUES ('C000000002', 2);

INSERT
INTO Contain (commentID, postID)
VALUES ('C000000003', 3);

INSERT
INTO Contain (commentID, postID)
VALUES ('C000000004', 4);


INSERT
INTO Contain (commentID, postID)
VALUES ('C000000005', 5);
```

TABLE AccountOwns

```
INSERT
INTO AccountOwns (userName, displayName, numFollowing, numFollowers, numPosts, userID)
VALUES ('userOne', 'User One', 100, 150, 50, 1);

INSERT
INTO AccountOwns (userName, displayName, numFollowing, numFollowers, numPosts, userID)
VALUES ('userTwo', 'User Two', 200, 250, 150, 2);

INSERT
INTO AccountOwns (userName, displayName, numFollowing, numFollowers, numPosts, userID)
VALUES ('userThree', 'User Three', 300, 350, 250, 3);

INSERT
INTO AccountOwns (userName, displayName, numFollowing, numFollowers, numPosts, userID)
VALUES ('userFour', 'User Four', 400, 450, 350, 4);
```

```
INSERT
INTO AccountOwns (userName, displayName, numFollowing, numFollowers, numPosts, userID)
VALUES ('userFive', 'User Five', 500, 550, 450, 5);
```

TABLE PublishPosts

```
INSERT
INTO PublishPosts (PostID, location, postDate, userID)
VALUES (1, 'New York', '2024-03-01', 1);

INSERT
INTO PublishPosts (PostID, location, postDate, userID)
VALUES (2, 'Los Angeles', '2024-03-02', 2);

INSERT
INTO PublishPosts (PostID, location, postDate, userID)
VALUES (3, New York', '2024-03-03', 3);

INSERT
INTO PublishPosts (PostID, location, postDate, userID)
VALUES (4, 'Miami', '2024-03-04', 4);

INSERT
INTO PublishPosts (PostID, location, postDate, userID)
VALUES (5, New York', '2024-03-05', 5);
```