1. **A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.**

A social media platform where users can join different communities to post, comment, participate in events, and give awards. This is modeling what a typical social media platform would look like. Alongside the basic social media functions, admins are entities that can review tickets created by users.

2. **The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.**
   We're using a similar ER diagram because we didn't receive feedback on our first.

   We changed attributes date and time to dateTime in events to match SQL type

   We changed the participation constraint on Comments to CommentsOn from partial to total participation.

   We changed the participation constraint on Posts from partial to total participation.

   We added various attributes to Users, Video, Communities, Admins to model more functionality.

3. **The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:**
    a. *List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.*
    b. *Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.*

Users(*userName*: VARCHAR[255], *biography:* CLOB, *firstName*: VARCHAR[255], *birthDay*: INT, *birthMonth*: VARCHAR[16], *birthYear*: YEAR, *starSign*: VARCHAR[255])

- firstName: NOT NULL
- birthDay: NOT NULL
- birthMonth: NOT NULL
- birthYear: NOT NULL
- starSign: NOT NULL
- username: DEFAULT("Unknown User")

Follows(***userName***: VARCHAR[255], ***followsUserName***: VARCHAR[255])

- userName: NOT NULL
- followsUserName: NOT NULL

Attends(***userName***: VARCHAR[255], ***eventName***: VARCHAR[255])

- userName: NOT NULL
- eventName: NOT NULL

MemberOf(***userName***: VARCHAR[255], ***communityName***: VARCHAR[255])

- communityName: NOT NULL
- userName: NOT NULL

Posts(*postId*: VARCHAR[36], *title*: VARCHAR[255], ***userName***: VARCHAR[255], ***communityName***: VARCHAR[255], *dateTime*: TIMESTAMP)

- userName: NOT NULL
- communityName: NOT NULL
- dateTime: NOT NULL
- UNIQUE(userName, dateTime)
- CANDIDATE KEY: {userName, dateTime}

Image(***postId***: VARCHAR[36], *caption*: VARCHAR[255])

Textpost(***postId***: VARCHAR[36], *text*: CLOB)

- text: NOT NULL

Video(***postId***: VARCHAR[36], *caption*: VARCHAR[255], *duration*: INT, *size*: VARCHAR[255])

- duration: NOT NULL

- size: NOT NULL

Communities(*name*: VARCHAR[255], *genre*: VARCHAR[255], *description*: CLOB, *capacity*: INT, *ageRange*: VARCHAR[255])

- genre: DEFAULT("General")
- capacity: NOT NULL
- ageRange: NOT NULL

Events(*eventName:* VARCHAR[255], *dateTime*: TIMESTAMP, *location*: VARCHAR[255], ***communityName*:** VARCHAR[255])

- communityName: NOT NULL
- eventName: NOT NULL

Help Tickets(*ticketId*: VARCHAR[36], *description*: CLOB, *type*: VARCHAR[255], ***userName*:** VARCHAR[255], ***adminId*:** VARCHAR[36])

- userName: NOT NULL
- description: NOT NULL
- type: NOT NULL

Admins(*adminID*: VARCHAR[36], *name*: VARCHAR[255], *experienceLevel*: VARCHAR[255])

- name: NOT NULL
- experienceLevel: NOT NULL

Comments(*commentId*: VARCHAR[36], *text*: CLOB, ***userName***: VARCHAR[255], ***postId***: VARCHAR[36], *dateTime*: TIMESTAMP)

- text: NOT NULL
- userName: NOT NULL
- postID: NOT NULL
- dateTime: NOT NULL
- UNIQUE(userName, dateTime)
- CANDIDATE KEY: {userName, dateTime}

Awards(*type*: VARCHAR[255], ***postId*:** VARCHAR[36], *cost*: FLOAT)

4. **Functional Dependencies (FDs)**
    a. *Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key).*
    PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A → A.

    ***Note:** In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process.*

Users(userName, biography, firstName, birthDay, birthMonth, birthYear, starSign)
- userName → biography,firstName,birthDay,birthMonth,birthYear,starSign
- birthMonth,birthYear → starSign

Follows(**userName**, **followsUserName**)

Attends(**userName**, **eventName**)

MemberOf(**userName**, **communityName**)

Posts(postId, title, **userName**, **communityName,** dateTime)
- postId → title,userName,communityName,dateTime

Image(**postId**, caption)
- postId → caption

Textpost(**postId**, text)
- postId → text

Video(**postId**, caption, duration, size)
- postId → caption,duration,size
- duration → size

Communities(name, genre, description, capacity, ageRange)
- name → genre,description,capacity,ageRange
- genre → capacity
- genre → ageRange

Events(eventName, dateTime, location, **communityName**)
- eventName,dateTime → location,communityName

Help Tickets(ticketId, description, type, **userName, adminId**)

- ticketId → description,type,userName,adminId

Admins(adminId, name, experienceLevel)

- adminId → name,experienceLevel

Comments(commentId, text, **userName**, **postId**, dateTime)

- commentId → text, userName, postId, dateTime

Awards(type, **postId,** cost)

- type → cost

5. **Normalization**
    a. *Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post normalization.*

DECOMPOSITION STEPS

Users(<u>userName</u>, biography, firstName, birthDay, birthMonth, birthYear, starSign)

Users (userName, biography, FirstName, day, month, year, starSign)

FDs: ① userName → biography, firstName, day, month, year, starSign

② day, month → starSign

Users is not in BCNF b/c of FD ② ({day, month} is not a superkey)

Decompose: (userName, biography, firstName, year) (day, month) (starSign)

R_1 (day, month, starSign) R_2 (userName, biography, FirstName, day, month, year)

R_1 is in BCNF ({day, month} is a superkey)
R_2 is in BCNF (no FDs apply)

StarSigns (<u>day, month</u>, starSign)
Users (<u>userName</u>, biography, FirstName, day, month, year)

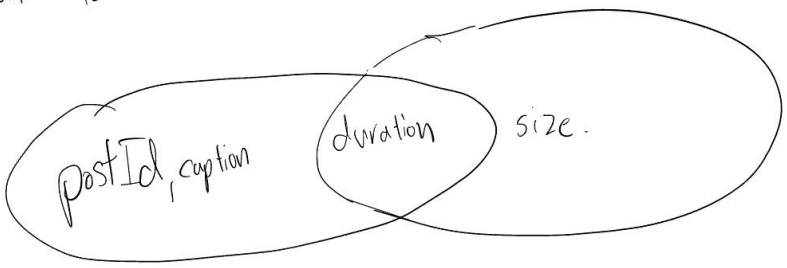
Video(**postId**, caption, duration, size)

Video (postId , caption , duration , size)

FDs: postId → caption, duration, size
duration → size

duration is not a super key

Decompose:



R1 (caption, postId, duration) R2 (duration, size)

R1 is in BCNF
R2 is in BCNF

Videos (postId , caption, duration) StorageSizes (duration , size)

Communities(name, genre, description, capacity, ageRange)

Communities(name, genre, description, capacity, ageRange)

FDs:

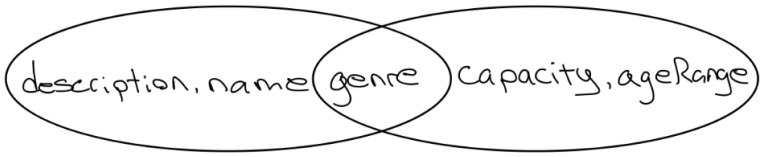
1. name → genre, description, capacity, ageRange
2. genre → capacity
3. genre → ageRange

$name^+ = \{name, genre, description, capacity, ageRange\}$

$genre^+ = \{genre, capacity, ageRange\}$

Genre is not a superkey so Communities is not in BCNF

genre → capacity, ageRange



R_1(genre, capacity, ageRange) R_2(description, name, genre)

genreDetails(genre, capacity, ageRange)

communityInfo(name, description, genre)

FINAL TABLES

Users(*userName*: VARCHAR[255], *biography:* CLOB, *firstName*: VARCHAR[255], ***birthDay***: INT, ***birthMonth***: VARCHAR[16], *birthYear*: YEAR)

- firstName: NOT NULL
- birthDay: NOT NULL
- birthMonth: NOT NULL
- birthYear: NOT NULL
- username: DEFAULT("Unknown User")

StarSigns(birthDay: INT, birthMonth: VARCHAR[16], starSign: VARCHAR[255])

- starSign: NOT NULL

Follows(***userName***: VARCHAR[255], ***followsUserName***: VARCHAR[255])

- userName: NOT NULL
- followsUserName: NOT NULL

Attends(***userName***: VARCHAR[255], ***eventName***: VARCHAR[255])

- userName: NOT NULL
- eventName: NOT NULL

MemberOf(***userName***: VARCHAR[255], ***communityName***: VARCHAR[255])

- communityName: NOT NULL
- userName: NOT NULL

Posts(*postId*: VARCHAR[36], *title*: VARCHAR[255], ***userName***: VARCHAR[255], ***communityName***: VARCHAR[255], *dateTime*: TIMESTAMP)

- userName: NOT NULL
- communityName: NOT NULL
- dateTime: NOT NULL
- UNIQUE(userName, dateTime)
- CANDIDATE KEY: {userName, dateTime}

Image(***postId***: VARCHAR[36], *caption*: VARCHAR[255])

- postId: NOT NULL

Textpost(***postId***: VARCHAR[36], *text*: CLOB)

- postId: NOT NULL
- text: NOT NULL

Video(***postId***: VARCHAR[36], *caption*: VARCHAR[255], ***duration***: INT)

- postId: NOT NULL
- duration: NOT NULL

StorageSizes(duration: INT, size: VARCHAR[255])

- size: NOT NULL

Communities(*name*: VARCHAR[255], *description*: CLOB, ***genre***: VARCHAR[255])

- genre: DEFAULT("General")

Genres(*genre*: VARCHAR[255], *capacity*: INT, *ageRange*: VARCHAR[255])

- capacity: NOT NULL
- ageRange: NOT NULL

Events(*eventName*: VARCHAR[255], *dateTime*: TIMESTAMP, *location*: VARCHAR[255], ***communityName*:** VARCHAR[255])

- communityName: NOT NULL
- eventName: NOT NULL

Help Tickets(*ticketId*: VARCHAR[36], *description*: CLOB, *type*: VARCHAR[255], ***userName*:** VARCHAR[255], ***adminId*:** VARCHAR[36])

- userName: NOT NULL
- description: NOT NULL
- type: NOT NULL

Admins(*adminID*: VARCHAR[36], *name*: VARCHAR[255], *experienceLevel*: VARCHAR[255])

- name: NOT NULL
- experienceLevel: NOT NULL

Comments(*commentId*: VARCHAR[36], *text*: CLOB, ***userName***: VARCHAR[255], ***postId***: VARCHAR[36], *dateTime*: TIMESTAMP)

- text: NOT NULL
- userName: NOT NULL
- postID: NOT NULL
- dateTime: NOT NULL
- UNIQUE(userName, dateTime)
- CANDIDATE KEY: {userName, dateTime}

Awards(*type*: VARCHAR[255], ***postId*:** VARCHAR[36], *cost*: FLOAT)

6. **The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc.**

```
CREATE TABLE Genres(
      genre VARCHAR(255) PRIMARY KEY,
      capacity INTEGER NOT NULL,
      ageRange VARCHAR(255) NOT NULL
);

CREATE TABLE Communities(
      name VARCHAR(255) PRIMARY KEY,
      description CLOB NOT NULL,
      genre VARCHAR(255) NOT NULL DEFAULT 'General'
);

CREATE TABLE Users(
      userName VARCHAR(255) PRIMARY KEY,
      biography CLOB,
      firstName VARCHAR(255) NOT NULL,
      birthDay INTEGER NOT NULL,
      birthMonth VARCHAR(16) NOT NULL,
      birthYear YEAR NOT NULL,
      FOREIGN KEY (birthDay, birthMonth) REFERENCES StarSigns(birthDay,
birthMonth)
);

CREATE TABLE StarSigns(
      birthDay INTEGER,
      birthMonth VARCHAR(16),
      starSign VARCHAR(255) NOT NULL,
      PRIMARY KEY(birthDay, birthMonth)
);
```

```
CREATE TABLE Follows(
      userName VARCHAR(255),
      followsUserName VARCHAR(255),
      PRIMARY KEY(userName, followsUserName),
      FOREIGN KEY(userName) REFERENCES Users(userName)
      FOREIGN KEY(followsUserName) REFERENCES Users(userName)
);

CREATE TABLE Attends(
      userName VARCHAR(255),
      eventName VARCHAR(255),
      PRIMARY KEY(userName, eventName),
      FOREIGN KEY(userName) REFERENCES Users(userName)
      FOREIGN KEY(eventName) REFERENCES Events(eventName)
);

CREATE TABLE MemberOf(
      userName VARCHAR(255),
      communityName VARCHAR(255),
      PRIMARY KEY(userName, communityName),
      FOREIGN KEY(userName) REFERENCES Users(userName)
      FOREIGN KEY(communityName) REFERENCES Communities(name)
);

CREATE TABLE Posts(
      postId CHAR(36) PRIMARY KEY,
      title VARCHAR(255),
      userName VARCHAR(255) NOT NULL DEFAULT 'Unknown User',
      communityName VARCHAR(255) NOT NULL,
      dateTime TIMESTAMP NOT NULL,
      UNIQUE(userName, dateTime),
      FOREIGN KEY(userName) REFERENCES Users(userName) ON DELETE SET
DEFAULT
      FOREIGN KEY(communityName) REFERENCES Communities(name)
);
```

```
CREATE TABLE Image(
      postId CHAR(36) PRIMARY KEY,
      caption VARCHAR(255),
      FOREIGN KEY(postId) REFERENCES Posts(postId) ON DELETE CASCADE
);

CREATE TABLE Textpost(
      postId CHAR(36) PRIMARY KEY,
      text CLOB NOT NULL,
      FOREIGN KEY(postId) REFERENCES Posts(postId) ON DELETE CASCADE
);

CREATE TABLE Video(
      postId CHAR(36) PRIMARY KEY,
      caption VARCHAR(255),
      duration INT NOT NULL,
      FOREIGN KEY (duration) REFERENCES StorageSizes(duration)
);

CREATE TABLE StorageSizes(
      duration INT PRIMARY KEY,
      size VARCHAR(255) NOT NULL
);


CREATE TABLE Events(
      eventName VARCHAR(255),
      dateTime TIMESTAMP NOT NULL,
      location VARCHAR(255),
      communityName VARCHAR(255) NOT NULL,
      PRIMARY KEY (eventName, dateTime),
      FOREIGN KEY (communityName) REFERENCES Communities(name)
);
```

```
CREATE TABLE HelpTickets(
      ticket CHAR(36) PRIMARY KEY,
      description CLOB NOT NULL,
      type VARCHAR(255) NOT NULL,
      userName VARCHAR(255) NOT NULL,
      adminId CHAR(36),
      FOREIGN KEY (userName) REFERENCES Users(userName),
      FOREIGN KEY (adminId) REFERENCES Admins(adminID),
);

CREATE TABLE Admins(
      adminID CHAR(36) PRIMARY KEY,
      name VARCHAR(255) NOT NULL,
      experienceLevel VARCHAR(255) NOT NULL
);

CREATE TABLE Comments(
      commentId CHAR(36) PRIMARY KEY,
      userName VARCHAR(255) DEFAULT 'Unknown',
      postId VARCHAR(255),
      text CLOB NOT NULL,
      dateTime TIMESTAMP NOT NULL,
      FOREIGN KEY (userName) REFERENCES Users(userName) ON DELETE SET
DEFAULT,
      FOREIGN KEY (postId) REFERENCES Posts(postId) ON DELETE CASCADE,
      UNIQUE (userName, dateTime)
);

CREATE TABLE Awards(
      type VARCHAR(255) ,
      postId CHAR(36),
      cost FLOAT NOT NULL,
      PRIMARY KEY (postId, type),
      FOREIGN KEY (postId) REFERENCES Posts(postId) ON DELETE CASCADE
);
```

7. **INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.**
    a. *Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.*

Genres

```
INSERT INTO Genres VALUES('Sports', 1000, '12-30');
INSERT INTO Genres VALUES('Health', 200, '19-60');
INSERT INTO Genres VALUES('Gaming', 200, '11-19');
INSERT INTO Genres VALUES('Finance', 200, '20-45');
INSERT INTO Genres VALUES('Tech', 200, '19-30');
```

Communities

```
INSERT INTO Communities VALUES('Stocks','The top community for all
things stock related.', 'Finance');
INSERT INTO Communities VALUES('Gym','Big gains', 'Health');
INSERT INTO Communities VALUES('Minecraft','Minecraft community for
building', 'Gaming');
INSERT INTO Communities VALUES('LiverpoolFC','The community for the
reds', 'Sports');
INSERT INTO Communities VALUES('Databases','Nerdiest Community in
Town', 'Computer Science');
```

Users

```
INSERT INTO Users VALUES('dman','Cool Bio Alert', 'Daven', 12,
'January', 2003);
INSERT INTO Users VALUES('304ta','im a cool TA!', 'Joe', 10,
'November', 2003);
INSERT INTO Users VALUES('sevaman','Worlds Best Prof', 'Seva', 8,
'September', 1990);
INSERT INTO Users VALUES('olti123','the best group member', 'Olti', 9,
'August', 2003);
INSERT INTO Users VALUES('rickybangbang', 'elite engineer', 'Ricky',
7, 'April', 2003);
```

StarSigns

```
INSERT INTO StarSigns VALUES(12,'January', 'Capricorn');
INSERT INTO StarSigns VALUES(10,'November', 'Scorpio');
INSERT INTO StarSigns VALUES(9,'August', 'Leo');
INSERT INTO StarSigns VALUES(8,'September', 'Virgo');
INSERT INTO StarSigns VALUES(7,'April', 'Aries');
```

Follows

```
INSERT INTO Follows VALUES('rickybangbang','sevaman');
INSERT INTO Follows VALUES('olti123','sevaman');
INSERT INTO Follows VALUES('304ta','sevaman');
INSERT INTO Follows VALUES('sevaman','olti123');
INSERT INTO Follows VALUES('dman','rickybangbang');
```

Attends

```
INSERT INTO Attends VALUES('olti123','Workout Day');
INSERT INTO Attends VALUES('rickybangbang','Christmas Stock
Celebration');
INSERT INTO Attends VALUES('304ta','Minecraft Thanksgiving Event');
INSERT INTO Attends VALUES('sevaman','SQL Party');
INSERT INTO Attends VALUES('rickybangbang','Cardio Hating Party');
```

MemberOf

```
INSERT INTO MemberOf VALUES('rickybangbang','Stocks');
INSERT INTO MemberOf VALUES('olti123','Gym');
INSERT INTO MemberOf VALUES('304ta','Minecraft');
INSERT INTO MemberOf VALUES('sevaman','Databases');
INSERT INTO MemberOf VALUES('rickybangbang','Gym');
```

Posts

```
INSERT INTO Posts VALUES(1, 'Hello World', 'sevaman', 'Databases',
'2023-10-20 23:59:59');
INSERT INTO Posts VALUES(2, 'Investing is Awesome!', 'rickybangbang',
'Stocks', '2023-08-01 10:30:12');
INSERT INTO Posts VALUES(3, 'How to get strong', 'olti123', 'Gym',
'2023-11-01 14:21:00');
INSERT INTO Posts VALUES(4, 'Minecraft Lets Play', '304ta',
'Minecraft', '2023-10-08 03:00:00');
INSERT INTO Posts VALUES(5, 'How to get rich', 'rickybangbang', 'Gym',
'2023-12-31 22:05:24');
```

```
INSERT INTO Posts VALUES(6, ‘Databases 101’, ‘sevaman’, ‘Databases’,
'2023-10-02 20:49:12');
INSERT INTO Posts VALUES(7, ‘Stocks for dummies’, ‘rickybangbang’,
‘Stocks’, '2023-05-01 14:30:20');
INSERT INTO Posts VALUES(8, ‘What is the gym?’, ‘olti123’, ‘Gym’,
'2023-09-20 07:01:12’);
INSERT INTO Posts VALUES(9, ‘What’s your favourite mob?’, ‘304ta’,
‘Minecraft’, '2023-11-10 05:21:28');
INSERT INTO Posts VALUES(10, ‘Working out at the gym for 24 hours
challenge’, ‘rickybangbang’, ‘Gym’, '2022-01-01 21:00:21');
INSERT INTO Posts VALUES(11, ‘CPSC 304 Is The Best’, ‘sevaman’,
‘Databases’, '2023-04-02 21:30:21');
INSERT INTO Posts VALUES(12, ‘S&P 500 Is Up!’, ‘rickybangbang’,
‘Stocks’, '2023-08-01 10:30:12');
INSERT INTO Posts VALUES(13, ‘Bicep Curls are for babies’, ‘olti123’,
‘Gym’, '2023-09-02 08:32:40');
INSERT INTO Posts VALUES(14, ‘My fav block review’, ‘304ta’,
‘Minecraft’, '2022-08-08 08:08:08');
INSERT INTO Posts VALUES(15, ‘My legs are sore’, ‘rickybangbang’,
‘Gym’, '2023-11-30 21:08:01');
```

Image

```
INSERT INTO Image VALUES(1, 'This is my first post! Enjoy the photo
:D');
INSERT INTO Image VALUES(2, 'Look at this stack of cash I made
investing!');
INSERT INTO Image VALUES(9, 'Look at this minecraft pig!');
INSERT INTO Image VALUES(11, '3NF is my favourite');
INSERT INTO Image VALUES(15, 'IM HUUUUGE');
```

Textpost

```
INSERT INTO Textpost VALUES(3, 'Here’s a step by step guide on how to
get strong. Step 1. Get strong!');
INSERT INTO Textpost VALUES(7, 'For all you dummies out there, listen
up');
INSERT INTO Textpost VALUES(8, 'Today i'm gonna talk about what is the
gym.');
INSERT INTO Textpost VALUES(12, 'Check the market! Invest now!');
```

```
INSERT INTO Textpost VALUES(13, 'If you do bicep curls ur the
worst!');
```

Video

```
INSERT INTO Video VALUES(4, 'Welcome back to my Minecraft video!',
120);
INSERT INTO Video VALUES(5, 'Get rich with these quick tips!', 30);
INSERT INTO Video VALUES(6, 'Watch to learn all about SQL!', 200);
INSERT INTO Video VALUES(10, 'I almost fainted!', 600);
INSERT INTO Video VALUES(14, 'Number one will surprise you!', 1200);
```

StorageSizes

```
INSERT INTO StorageSizes VALUES(120, ‘Medium’);
INSERT INTO StorageSizes VALUES(30, 'Small');
INSERT INTO StorageSizes VALUES(200, ‘Medium’);
INSERT INTO StorageSizes VALUES(600, 'Large’);
INSERT INTO StorageSizes VALUES(1200, ‘Large’);
```

Events

```
INSERT INTO Events VALUES(‘Workout Day’, ‘2024-10-31’, ‘UBC’, ‘Gym’);
INSERT INTO Events VALUES(‘Christmas Stock Celebration’, ‘2024-12-25’,
‘UBC’, ‘Stocks’);
INSERT INTO Events VALUES(‘Minecraft Thanksgiving Event’, ‘2024-10-09,
‘UBC’, ‘Minecraft’);
INSERT INTO Events VALUES(‘SQL Party’, ‘2024-01-31’, ‘UBC’,
‘Databases’);
INSERT INTO Events VALUES(‘Cardio Hating Party’, ‘2024-03-01’, ‘UBC’,
‘Gym’);
```

Help Tickets

```
INSERT INTO HelpTickets VALUES(1, ‘Spammed Craft, ‘Spam’, ‘dman’, 1);
INSERT INTO HelpTickets VALUES(2, ‘Harassed’, ‘Harass’, ‘olti123’, 1);
INSERT INTO HelpTickets VALUES(3, ‘Racist Post’, ‘Racism’, ‘sevaman’,
1);
INSERT INTO HelpTickets VALUES(4, ‘Broke a Comment’, ‘Bug’, ‘sevaman’,
1);
INSERT INTO HelpTickets VALUES(5, ‘Ban Water’, ‘Ban’, ‘olti123’, 1);
```

Admins

```
INSERT INTO Admins VALUES(1, ‘james’, ‘King’)
INSERT INTO Admins VALUES(2, ‘elizabeth’, ‘Queen’)
INSERT INTO Admins VALUES(3, ‘andjill’, ‘Jack’)
INSERT INTO Admins VALUES(4, ‘littlemonkeys’, ‘Ten’)
INSERT INTO Admins VALUES(5, ‘tails’, ‘Nine’)
```

Comments

```
INSERT INTO Comments VALUES(1, ‘Daven is smart’, ‘olti123’, 1,
‘2024-12-24 22:05:24’)
INSERT INTO Comments VALUES(2, ‘Daven is not smart’, ‘rickybangbang’,
1, ‘2024-12-25 22:05:25’)
INSERT INTO Comments VALUES(3, ‘Daven isn’t smart’, ‘olti123’, 1,
‘2024-12-26 22:05:26’)
INSERT INTO Comments VALUES(4, ‘Daven is very smart’, ‘sevaman’, 1,
‘2024-12-27 22:05:27’)
INSERT INTO Comments VALUES(5, ‘Daven is not so smart’, ‘olti123’, 1,
‘2024-12-28 22:05:28’)
```

Awards

```
INSERT INTO Awards VALUES(‘Gold’, 1, 1000.00)
INSERT INTO Awards VALUES(‘Silver’, 1, 500.00)
INSERT INTO Awards VALUES(‘Gold’, 2, 1000.00)
INSERT INTO Awards VALUES(‘Gold’, 3, 1000.00)
INSERT INTO Awards VALUES(‘Platinum’, 4, 5000.00)
```