1.

Category (CategoryID, CategoryName, Description)

Tags(TagName)

ActivityLogs(LogID, UserID, ThreadID, ViewTime)

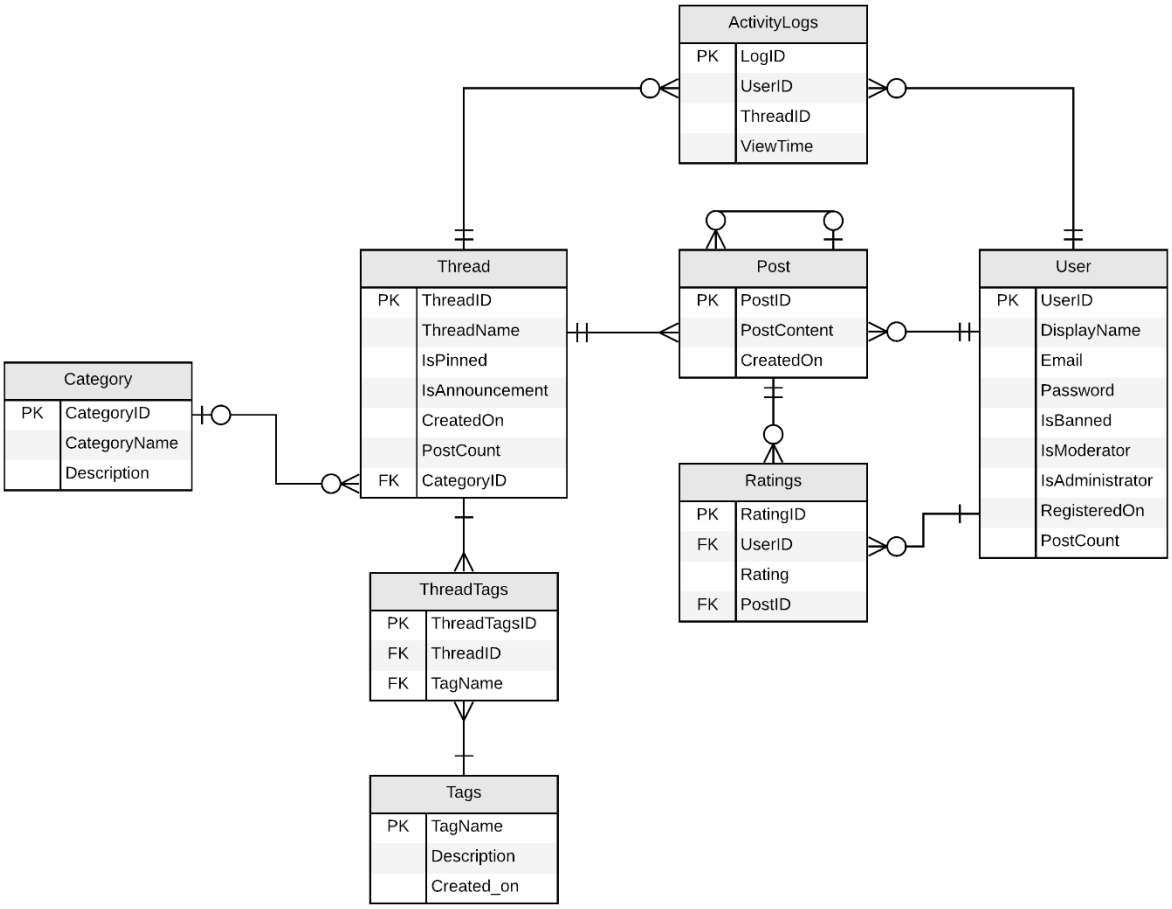
Ratings(RatingID, UserID, Rating)

Thread (ThreadID, ThreadName, IsPinned, IsAnnouncement, CreatedOn, PostCount)

Post (PostID, CreatedOn, ModifiedOn, Rating)

User(UserID,DisplayName,Email,Password,IsBanned,IsModerator,IsAdministrator,RegisteredOn, PostCount)

2.



3.

A composite key of post_id and user_id_rating

4.

a)

post_id -> topic_id

post_id -> topic_date

post_id -> post_date

post_id -> topic

post_id -> post

post_id -> category_id

post_id -> category

user_id_rating -> user_id

user_id_rating -> username

user_id_rating -> email

post_id,user_id_rating -> rating

b)

MasterTable(post_id,user_id_rating, topic_id,topic_date,post_date,topic post,category_id,category,rating,user_id,username,email)

splits into:

Post(PostID, TopicID, Topic_date, topic, Post_date, Post, CategoryID, category)

User(User_id, username, email, user_id_rating, rating)

5.

a)

within the post table:

post_id -> topic_id -> topic_date

post_id -> topic_id -> topic

post_id -> category_id -> category

within the post table:

user_id_rating -> user_id -> username

user_id_rating -> user_id -> email

b)

Post(PostID, Post_date, Post, rating)

Thread(ThreadID, ThreadName, IsPinned, IsAnnouncement, CreatedOn, PostCount, CategoryID)

Topic(TopicID, Topic_date, Topic)

Category(CategoryID, Category)

User(UserID, Username, Email)

Rating(RatingID, UserRatingID, Rating)

6.

```
-- Tags table
CREATE TABLE IF NOT EXISTS Tags (
Tagname TEXT PRIMARY KEY,
Description TEXT,
CreatedOn DATETIME
);

-- Rating Table
CREATE TABLE Rating (
RatingID INTEGER PRIMARY KEY,
UserRatingID INTEGER REFERENCES User (UserID),
PostID INTEGER REFERENCES Post (PostID),
Rating INTEGER CHECK( Rating IN(0,1,-1)),
FOREIGN KEY (UserRatingID)
        REFERENCES User(UserID)
                ON DELETE CASCADE,
FOREIGN KEY (PostID)
        REFERENCES Post (PostID)
                ON DELETE CASCADE
);

-- Category Table
CREATE TABLE IF NOT EXISTS Category (
CategoryID INTEGER PRIMARY KEY,
CategoryName TEXT NOT NULL UNIQUE,
Description TEXT
);
```

```
--Activity Log Table
CREATE TABLE IF NOT EXISTS ActivityLog (
LogID INTEGER PRIMARY KEY,
UserID INTEGER REFERENCES User (UserID),
ThreadID INTEGER REFERENCES Thread (ThreadID),
ViewTime DATETIME NOT NULL
);

--ThreadTags Table
-- DBeaver gave me some trouble enforcing the foreign key constraints so I wrote
-- them (redundantly) at the table and column level and it worked
PRAGMA foreign_keys = ON;
CREATE TABLE ThreadTags (
ThreadTagsID INTEGER PRIMARY KEY,
ThreadID INTEGER REFERENCES Thread (ThreadID),
TagName TEXT REFERENCES Tags (TagName),

FOREIGN KEY (ThreadID)
        REFERENCES Thread(ThreadID)
                ON DELETE CASCADE,

FOREIGN KEY (TagName)
        REFERENCES Tags(TagName)
                ON DELETE CASCADE
);
```

7.

```
BEGIN TRANSACTION;

--Category Table Data
INSERT INTO Category (
CategoryID,
CategoryName
)
SELECT DISTINCT CategoryID, CategoryName
FROM temp_table;

--Thread Table Data
INSERT INTO Thread (
ThreadID,
ThreadName,
CategoryID,
CreatedOn
)
SELECT DISTINCT topic_id, topic, CategoryID, topic_date
FROM temp_table tt

--User Table Data
INSERT INTO User (
UserID,
DisplayName,
Email
)
SELECT DISTINCT user_id, username, email
FROM temp_table tt
```

```
-- Rating Table Data
INSERT INTO Rating (
UserRatingID,
PostID,
Rating
)
SELECT DISTINCT user_id_rating, post_id, rating
FROM temp_table tt

-- Post Table Data
INSERT INTO Post (
PostID,
PostContent,
CreatedOn,
ThreadID,
AuthorUserID

)
SELECT DISTINCT post_id, post, post_date, topic_id, user_id
FROM temp_table tt

COMMIT;
```

8.

```
CREATE TRIGGER postModified
        AFTER UPDATE OF PostContent ON Post
        BEGIN
                UPDATE Post
                SET ModifiedOn = datetime(datetime('now', 'localtime'), 'utc')
                WHERE Post.PostID = NEW.PostID;
        END;
```

9.

```
--Insert trigger
CREATE TRIGGER newPost
        AFTER INSERT ON Post
        BEGIN
                UPDATE User
                SET PostCount = PostCount + 1
                WHERE User.UserID = NEW.AuthorUserID;

                UPDATE Thread
                SET PostCount = PostCount + 1
                WHERE Thread.ThreadID = NEW.ThreadID;
        END;
```

```
--Delete Trigger
CREATE TRIGGER deletedPost
	AFTER DELETE ON Post
	BEGIN
		UPDATE User
		SET PostCount = PostCount - 1
		WHERE User.UserID = NEW.AuthorUserID;

		UPDATE Thread
		SET PostCount = PostCount - 1
		WHERE Thread.ThreadID = NEW.ThreadID;
	END;
```

10.

```
CREATE TRIGGER deletedRating
	AFTER DELETE ON Rating
	BEGIN
		UPDATE Post
		SET Rating = Rating - OLD.rating
		WHERE Post.PostID = OLD.PostID;
	END;

CREATE TRIGGER newRating
	AFTER INSERT ON Rating
	BEGIN
		UPDATE Post
		SET Rating = Rating + NEW.rating
		WHERE Post.PostID = NEW.PostID;
	END;
```

```
CREATE TRIGGER updateRating
        AFTER UPDATE ON Rating
        BEGIN
                UPDATE Post
                SET Rating = Rating + NEW.rating
                WHERE Post.PostID = NEW.PostID;
        END;
```