Team no:12

Project Phase –1 (DNA)

Social Media Platform Mini-World

This database is designed to support a social media platform, where users can create profiles, interact through posts, comments, messages, and participate in various activities. Each user can perform social media actions, such as posting updates, commenting on posts, replying to comments, and creating or joining groups. The platform accommodates verified users by tracking verification methods and timestamps, providing a higher trust level for interactions and connections. Additionally, users can organize and attend events, manage or follow business and community pages, and maintain relationships through friend connections

Purpose of the DataBase:

The purpose of the database is to enable efficient management of user-generated content, community interactions, and social connections, ensuring that data integrity is maintained across the platform's features

Users of the Database:

Platform Administrators:

These users will utilize the database to manage user registrations, verify accounts, and oversee the platform's overall operation. They ensure data accuracy and security by monitoring users' interactions, managing group memberships, and handling issues with posts, comments, or pages.

Regular Users

Basic account holders,Users who participate in friend relationships,People who follow pages,people who join groups as members,people who attend events,People who react to content (posts/comments/replies)

Content Moderators:

Moderators access the database to track, review, and moderate posts, comments, and replies. They handle flagged content, enforce community guidelines, and manage reactions on posts to maintain a safe and inclusive environment for users.

Event Organizers and Page Managers:

Users who create events or manage pages will use the database to set up and manage details related to events, such as participant lists and schedules. They can also manage their follower base, update page details, and interact with page followers.

Uses of the Database

General Users: General users would use the database to manage and update their profiles, connect with friends, post content, and participate in discussions. They could create posts, leave comments, reply to others, react to content, and organize or join groups and events.

Content Moderators: Moderators would use the database to manage content quality and enforce community guidelines. They would review flagged content (e.g., inappropriate posts or comments), manage interactions within groups.

Event Organizers and Page Managers: These users, often verified members or administrators, would interact with the database to plan, promote, and manage events. They would track participant counts, manage event schedules, and organize page content. Page managers might track follower engagement, post updates, and communicate with followers.

Platform Administrators: Administrators oversee the database structure and its users' data for accuracy, privacy, and security. They manage user verifications, handle user registrations, oversee database interactions, and monitor the system's health.

2. Database Requirements

User

Attributes:

- **UserID** *(Primary Key, Integer)*
- **Email** *(Primary key, Varchar)*

- **Username** *(Varchar, Unique, NOT_NULL)*
- **Full Name** *(Composite)*:
    - **FirstName** *(Varchar)*
    - **Middle Name** *(Varchar, optional)*
    - **LastName** *(Varchar)*
- **Birthdate** *(Date: DD-MM-YYYY)*
- **Gender**: *Enum [‘Male’, ‘Female’, ‘Other’] (Single-valued)*
- **Phone Numbers** *(Multivalued: Array of Varchar)*
- **Age** *(Int, Derived: From Birthdate)*
- **CreatedAt** *(Timestamp, Derived)*

SUB CLASS: VERIFIED USER

Attributes:

- **UserID** *(Primary Key, Integer)*
- **Email** *(Primary key, Varchar)*
- **Username** *(Varchar, Unique, NOT_NULL)*
- **Full Name** *(Composite)*:
    - **FirstName** *(Varchar)*
    - **Middle Name** *(Varchar, optional)*
    - **LastName** *(Varchar)*
- **Birthdate** *(Date)*
- **Gender**: *Enum [‘Male’, ‘Female’, ‘Other’] (Single-valued)*
- **Phone Numbers** *(Multivalued: Array of Varchar)*
- **Age** *(Int, Derived: From Birthdate)*
- **CreatedAt** *(Timestamp, Derived)*
- **VerificationMethod** (Enum [‘Email’, ‘Phone’, ‘GovID’]):
- **VerifiedAt** (Timestamp):

Post

Attributes:

- **PostID** *(Primary Key, Integer)*
- **Content** *(Composite)*:
    - **Text** *(Text)*

    - **MediaLinks** *(Array of Varchar, optional)*
- **Visibility** *(Enum ['Private', 'Friends-only', 'Public'])*
- **CreatedAt** *(Timestamp)*
- **ModifiedAt** *(Timestamp)*
- *Posted_by_userid(Foriegn key:Integer)*

Comment

Attributes:

- **Commented_PostID** *(Foriegn key, Integer)*
- **CommentID** *(Primary Key,Int)*
- ***Commented_USERID*** *(Foriegn key, Integer)*
- **Content** *(Text)*
- **CreatedAt** *(Timestamp)*

Group

Attributes:

- **GroupID** *(Primary Key, Integer)*
- **GroupName** *(Varchar)*
- **GroupDescription** *(Text)*
- **Created_by_user_id** *(integer)*
- **CreatedAt** *(Timestamp)*

Event

Attributes:

- **EventID** *(Primary Key, Integer)*
- **EventName** *(Varchar)*
- **Description** *(Text)*
- **Location** *(Composite)*:
    - **Venue** *(Varchar)*
    - **City** *(Varchar)*
    - **Country** *(Varchar)*
- **StartTime** *(Timestamp)*

- **EndTime** *(Timestamp)*
- **CreatedBy_user_id**(Integer)
- **MaxParticipants** *(Derived: Integer)*

Message (Weak Entity to User)

Attributes:

- **SenderID** *(Foreign Key referencing User)*
- **ReceiverID** *(Foreign Key referencing User)*
- **Content** *(Composite)*:
    - **Text** *(Text)*
    - **MediaLink** *(Varchar, Optional)*
- **SentAt** *(Timestamp)*
- **SeenAt** *(Timestamp)*

Page (Weak Entity to User)

Attributes:

- **PageName** *(Varchar)*
- **Category** *(Enum [‘Business’, ‘Community’, etc.])*
- **FollowersCount** *(Derived: Integer from Friends_with relationship)*
- **Created_by_userid***(Foreign Key referencing User)*
- **CreatedAt** *(Timestamp)*
- **Page_content** *(Text)*

Reply

Attributes:

- **ReplyID** (*Primary Key, Int)*
- **Replied_UserID** *(Foreign Key referencing User)*
- **Replied_commentID** *(Foriegn key referencing User)*
- **Message** *(Text)*
- **SentAt** *(Timestamp)*
- **SeenAt** *(Timestamp, Nullable)*

Relationship Types

- **User - Creates - Post**
  - **Cardinality**: 1: N
  - Minmax:(0, N) -(1,1)
  - **Participation**: Total for Post, Partial for User
- **User – Gives – Comment**
  - **Cardinality**: 1: N
  - Minmax:(0, N) -(1,1)
  - **Participation**: Total for Comment, Partial for User
- **Post – Gets – Comment**
  - **Cardinality**: 1: N
  - Minmax:(0, N) -(1,1)
  - **Participation**: Total for Post, Partial for Comment
- **User - Friends With - User (Self-referencing, Friend With, Friend Of)**
  - **Cardinality**: M: N
  - Minmax:(0, M) - (0, N)
  - **Participation**: Partial for User, Total for Friendship
  - **Attributes**-user1_id, user2_id
- **User – Organizes – Event**
  - Attributes-user_id,event_id
  - Minmax:(0, N) -(1,1)
  - Cardinality: 1: N
  - Participation: Partial for User, Total for Event
- **Comment-gets-Reply**
  - **Cardinality**: 1: N
  - **Minmax:** (0, N) -(1,1)
  - **Participation**: Total for Reply, Partial for Comment
- **User - Member Of - Group**
  - **Cardinality**: M: N
  - **Minmax:** (0, M) -(1, N)
  - Attributes- user_id,group_id
  - **Participation**: Partial for User, Total for Group
- **User - Attends - Event**
  - **Cardinality**: M: N
  - **Minmax:** (0, M) -(0, N)
  - **Participation**: Total for Event, Partial for User

- **Reacts To -(User,Post,Comment,Reply)**
  - **Cardinality**: 1: N: K: P
  - **Minmax:** (0, M)-(0, N)-(0,K)-(0,P)
  - **Attributes-**user_id,post_id,comment_id,reply_id,reaction_type
  - **Participation**: Total for Reacts-To, Partial for User, Post, or Comment
- **Page - Managed By - User**
  - **Cardinality**: 1:1
  - **Minmax:** (1,1) -(0, 1)
  - **Participation**: Partial for User, Total for page
- **User - Follows – Page**
  - **Cardinality**: M: N
  - **Minmax:** (0, M) -(0, N)
  - **Attributes-**user_id,created_user_id
  - **Participation**: Total for Follows, Partial for User and Page
- **User – Communicates – Message**
  - **Cardinality**: 1: N
  - **Minmax:** (0, M) -(1, 1)
  - **Participation**: Partial for user, Total for message

Assumptions:

1) User can manage only a single page.
2) Replies don't have replies.
3) Users can create multiple posts.
4) Users can hold one event at a time.
5) The user who creates the page is not counted as follower.
6) The user who is the host of the event is not counted as the attendee.
7) The user who creates the group is counted as the member of the group.
8) Friendship relations are two-way.

3.) Functional Requirements

RETRIEVALS:

- **SELECTION:**
  - **GET_USERS_BY_GENDER**: Retrieve all the data of the users whose gender is male.
  - **GET_POSTS_BY_USER**: Retrieve all posts posted by user (using user_id).

- **GET_MESSAGES_BETWEEN_USERS:** Retrieve messages sent between two users (using user ids-UserID1 and UserID2).

- **PROJECTION**:
  - **GET_USERNAME_BY_AGE**: Get the names of the users whose is age is greater than 18
  - **GET_CONTENTOFPOST_BY_VISIBILITY**: Get the content of the posts whose visibility is 'public'.
  - **GET_REPLYID_OF_USER:** Get Reply IDs of replies made by a specific user (using user_id).
- **AGGREGATE**:
  - **COUNT_COMMENTS_BY_USERID:** Count of comments made by user using UserID .
  - **AVERAGE_FOLLOWERS_FOR_PAGES:** Calculates average number of followers across all pages.
- **SEARCH**:
  - **SEARCH_GROUPNAME_BY_WORD**: Search for groups whose name contains the substring "tech".
  - **SEARCH_EVENT_BY_WORD:** Search for all events where the Event Name contains the word **"conference"**

Analysis:

Analysis 1: "Average Number of Comments per Post by User Gender"

Objective: This report analyzes how user gender influences engagement in terms of comments on posts. It calculates the average number of comments per post for each gender.

Entities Involved:

- **User**
- **Post**
- **Comment**

We will calculate the total number of comments and posts made by users, grouped by gender. It uses joins to link users with posts and comments, then calculates the average

number of comments per post for each gender by dividing the total comments by the total posts.

Analysis 2: "User Age Distribution in Groups"

Objective: This report analyzes the age distribution of users within various groups, providing insights into the demographics participating in community discussions.

Entities Involved:

- **User**
- **Group**

We will list each group's name along with an age-range of users. It uses a join between groups and users, categorizes users into age ranges, counts the total users in each range for every group, and then organizes the output by group name and age range.

Analysis 3: "Average Response Time to Messages by User Gender"

Objective: This report analyzes the average time taken by users of different genders to respond to messages, providing insights into communication behaviors.

Entities Involved:

- **User**
- **Message**

We will calculate the average response time, in seconds, for seen messages, grouped by gender. It uses a join between messages and users, only considering messages that have been seen, and averages the time difference between when each message was sent and seen for each gender category.

MODIFICATIONS:

- **INSERT**:
    - INSERT INTO User (UserID, Email, Username, FirstName, MiddleName, LastName, Birthdate, Gender, Phone Numbers)
    - VALUES (1, 'nikhilesh@email.com', 'nikhilesht4115', 'Nikhilesh', 'Siva', 'Sai', '04-11-2005', 'Male', '["9032023829", "9087654321"]');

Checking for Violations:

Userid: Assuming that there is no user with user_id 1 to ensure that userid is unique. If there already exists a user with user_id =1 then it will cause violation.

Email: Since the email must be unique (because we mentioned it as primary key) and it must be of the form xx@email.com. In the above insertion the email ends with @gmail.com and assuming that there is no user with the given email if it is not unique then it causes violation.

Username: Since username must be unique. Assuming there is no user with the above username if the username already exists it will throw an error

Gender: Since the gender must be either male or female or other. In the above it is given male so there is no violation.

Phone numbers: Phone number must be 10 digits. Since both the phone numbers are of 10-digit length there will be no violation.

- **UPDATE**:
    - **Update the visibility of a post:** Update the visibility of the post with post_id=12 from public to private.

      Violations:

      Assuming that a post with post_id =12 exists, and its visibility is Public. Else it will cause an error.
    - **Update the Phone number of a User:** Update the 10-digit Phone number from '1234567890' to '9596816883' of the user with user_id =1

      Violations:

      Assuming that a user with user_id=1 exists, Else it will cause an error. Also since the given number is 10 digits, it is valid.
- **DELETE**:
    - **Delete_user_using_user_id :** Delete the user with the given user_id . As consequence of this operation, we must do the below operations also to not cause any violations
        - Delete all the posts created by the given user_id
        - Delete all the comments commented by the given user_id
        - Delete all the events created by the given user_id
        - Delete all the page created by the given user_id
        - Delete all the replies replied by the given user_id
        - Delete all the message that involves the given user_id

      We have to do all these operations to maintain referential integrity as all the above (entities) use the user_id as foreign key. Else we can simply ignore the above operations.

- **Delete group by Groupname :** Delete all the groups with given group name.
  Violations:
  Assuming that there exists at least one group with given name, else it will throw an error.