CSA0593 – DATABASE MANAGEMENT SYSTEM

PRAKALYA. PV

192311418

ASSIGNMENT- 4

**"Develop a database for a crowdfunding platform with projects, backers, donations, and reward tiers."  
  
-Model tables for crowdfunding projects, project backers, donation transactions, and reward tiers.**

**-Write stored procedures for creating a project, pledging donations, and distributing rewards.**

**-Implement triggers to update project funding status and notify backers of project milestones.**

**-Write SQL queries to find top backers, most-funded projects, and reward distribution rates**

Crowdfunding platforms are online tools that enable individuals or organizations to raise funds for their projects by appealing to a community of backers. A well-designed database is crucial for managing the interactions between projects, backers, and the platform. This database model aims to effectively manage data related to projects, users, donations, and reward tiers while ensuring scalability, reliability, and ease of reporting.

The following database schema captures the key entities and their relationships, ensuring a streamlined flow of information for functionalities like project creation, donation tracking, and reward management.

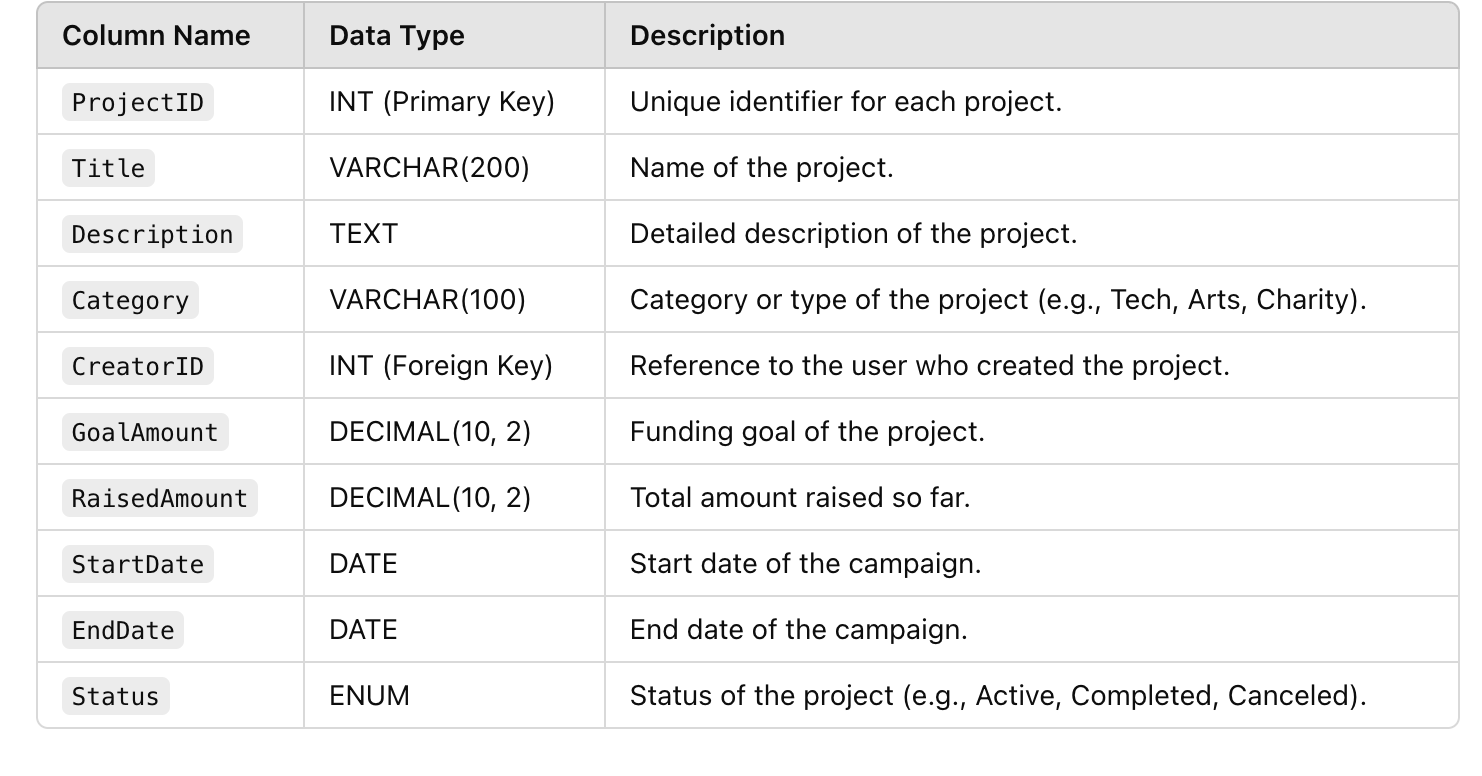
DATABASE STRUCTURE:



**-Model tables for crowdfunding projects, project backers, donation transactions, and reward tiers**:

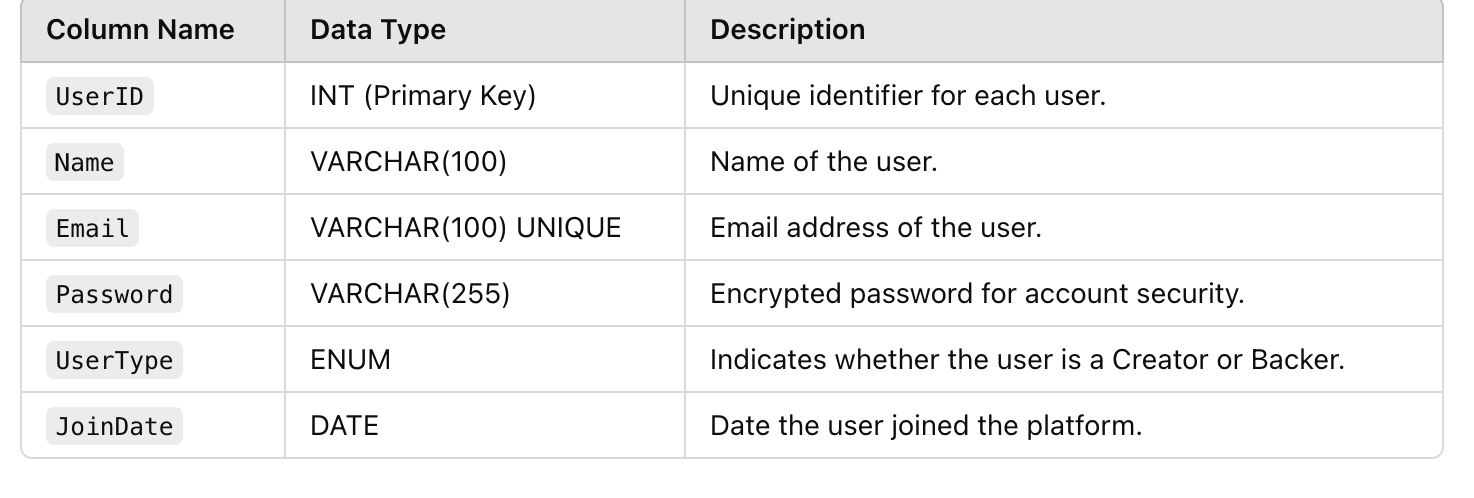
### ****1. Table: Projects****

This table stores information about each crowdfunding project.



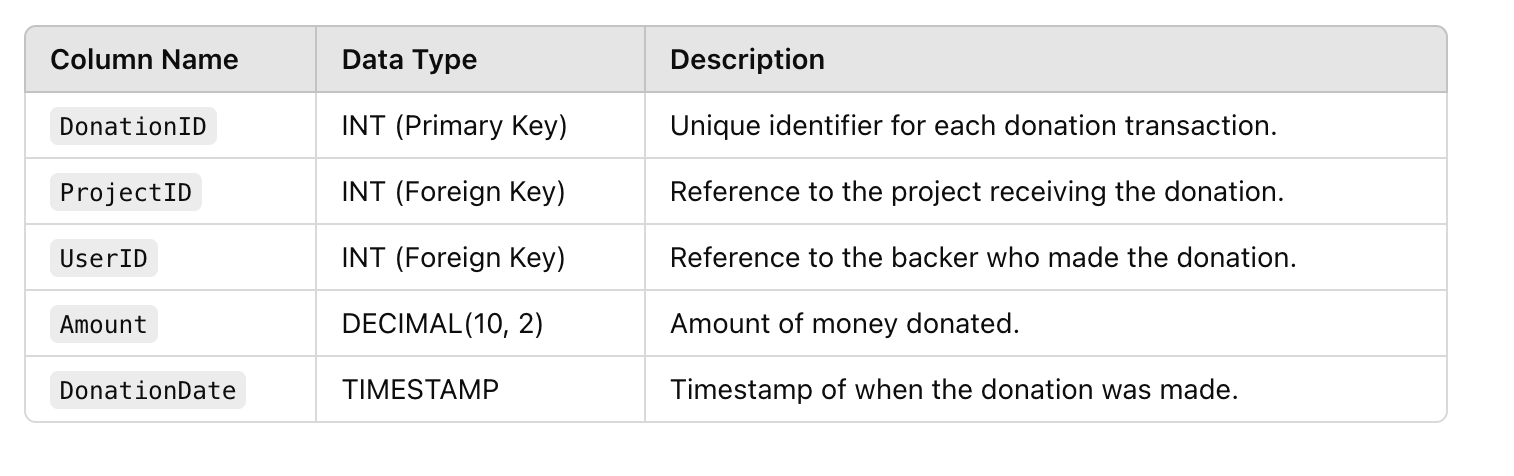
### ****2. Table: Users (Backers and Creators)****

This table holds data for both project backers and creators.



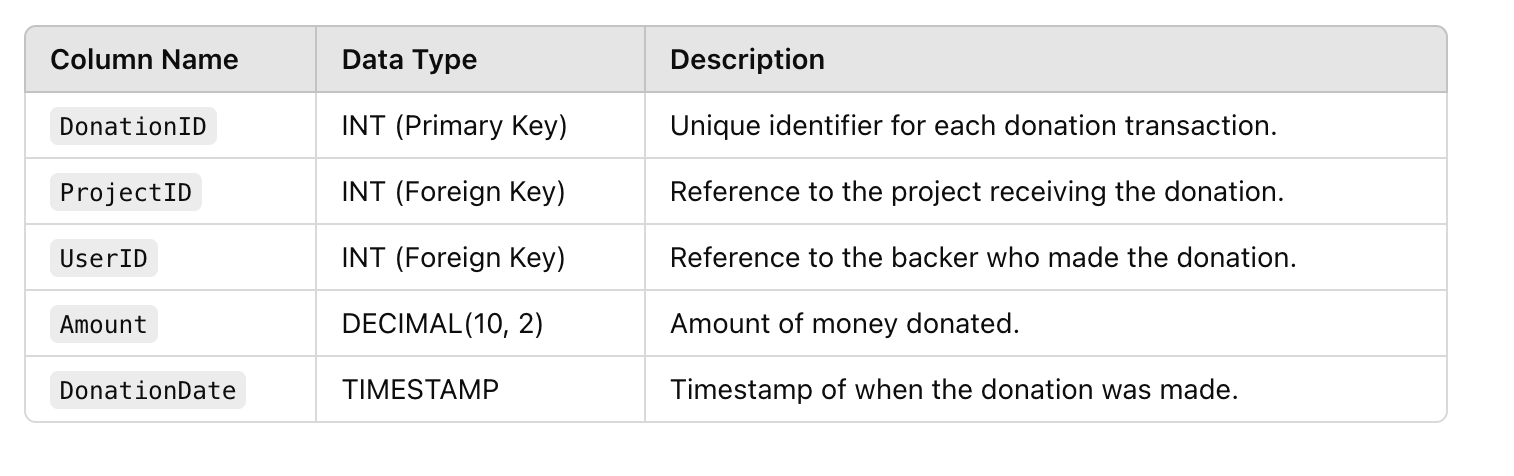
### ****3. Table: Donations****

This table records individual donation transactions for projects.



### ****4. Table: Reward Tiers****

This table tracks the reward tiers associated with each project.



### ****Relationships****

1. **Projects ↔ Users**:
   * A project is created by one user (creator), and each user can create multiple projects.
2. **Projects ↔ Donations**:
   * A project can have many donations, and each donation belongs to one project.
3. **Users ↔ Donations**:
   * A user (backer) can make multiple donations, and each donation is linked to one user.
4. **Projects ↔ Reward Tiers**:
   * A project can have multiple reward tiers, and each reward tier belongs to one project.

**-Write stored procedures for creating a project, pledging donations, and distributing rewards:**

**1. Creating a Project**

This stored procedure allows a user to create a new project by inserting its details into the Projects table

CREATE PROCEDURE CreateProject(

IN p\_Title VARCHAR(200),

IN p\_Description TEXT,

IN p\_Category VARCHAR(100),

IN p\_CreatorID INT,

IN p\_GoalAmount DECIMAL(10, 2),

IN p\_StartDate DATE,

IN p\_EndDate DATE

)

BEGIN

INSERT INTO Projects (Title, Description, Category, CreatorID, GoalAmount, StartDate, EndDate, Status)

VALUES (p\_Title, p\_Description, p\_Category, p\_CreatorID, p\_GoalAmount, p\_StartDate, p\_EndDate, 'Active');

END;

### ****2. Pledging Donations****

This stored procedure allows a user (backer) to pledge a donation to a project, updates the project's RaisedAmount, and increments the BackersCount in the appropriate reward tier (if applicable).

CREATE PROCEDURE PledgeDonation(

IN p\_UserID INT,

IN p\_ProjectID INT,

IN p\_Amount DECIMAL(10, 2),

IN p\_RewardID INT

)

BEGIN

INSERT INTO Donations (UserID, ProjectID, Amount, DonationDate)

VALUES (p\_UserID, p\_ProjectID, p\_Amount, NOW());

UPDATE Projects

SET RaisedAmount = RaisedAmount + p\_Amount

WHERE ProjectID = p\_ProjectID;

IF p\_RewardID IS NOT NULL THEN

UPDATE RewardTiers

SET BackersCount = BackersCount + 1

WHERE RewardID = p\_RewardID;

END IF;

END;

**3. Distributing Rewards**

This stored procedure distributes rewards to backers when a project is successfully funded. It identifies eligible backers based on their donations and the selected reward tiers.

CREATE PROCEDURE DistributeRewards(

IN p\_ProjectID INT

)

BEGIN

DECLARE finished INT DEFAULT 0;

DECLARE rewardID INT;

DECLARE amountRequired DECIMAL(10, 2);

DECLARE reward\_cursor CURSOR FOR

SELECT RewardID, AmountRequired

FROM RewardTiers

WHERE ProjectID = p\_ProjectID;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

OPEN reward\_cursor;

reward\_loop: LOOP

FETCH reward\_cursor INTO rewardID, amountRequired;

IF finished = 1 THEN

LEAVE reward\_loop;

END IF;

INSERT INTO RewardDistribution (RewardID, UserID)

SELECT rewardID, d.UserID

FROM Donations d

WHERE d.ProjectID = p\_ProjectID AND d.Amount >= amountRequired;

END LOOP;

CLOSE reward\_cursor;

END;

**-Implement triggers to update project funding status and notify backers of project milestones:**

### ****1. Trigger to Update Project Funding Status****

This trigger checks if a project's funding goal is met after a new donation and updates the project's Status column accordingly.

CREATE TRIGGER UpdateProjectStatus

AFTER INSERT ON Donations

FOR EACH ROW

BEGIN

DECLARE totalRaised DECIMAL(10, 2);

SELECT RaisedAmount

INTO totalRaised

FROM Projects

WHERE ProjectID = NEW.ProjectID;

IF totalRaised >= (SELECT GoalAmount FROM Projects WHERE ProjectID = NEW.ProjectID) THEN

UPDATE Projects

SET Status = 'Completed'

WHERE ProjectID = NEW.ProjectID;

END IF;

END;

**2. Trigger to Notify Backers of Project Milestones**

This trigger notifies backers when a project reaches specific funding milestones (e.g., 25%, 50%, 75%, 100%).

**Supporting Table for Notifications**

You may need a table to store notifications for backers:

CREATE TABLE Notifications (

NotificationID INT AUTO\_INCREMENT PRIMARY KEY,

UserID INT,

ProjectID INT,

Message TEXT,

NotificationDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (UserID) REFERENCES Users(UserID),

FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)

);

#### **Trigger Implementation**

The trigger calculates the percentage of the funding goal reached and inserts a notification when specific milestones are crossed.

CREATE TRIGGER NotifyMilestones

AFTER INSERT ON Donations

FOR EACH ROW

BEGIN

DECLARE fundingPercentage DECIMAL(5, 2);

DECLARE goalAmount DECIMAL(10, 2);

DECLARE raisedAmount DECIMAL(10, 2);

SELECT GoalAmount, RaisedAmount

INTO goalAmount, raisedAmount

FROM Projects

WHERE ProjectID = NEW.ProjectID;

SET fundingPercentage = (raisedAmount / goalAmount) \* 100;

IF fundingPercentage >= 25 AND fundingPercentage < 50 THEN

INSERT INTO Notifications (UserID, ProjectID, Message)

SELECT DISTINCT d.UserID, NEW.ProjectID, CONCAT('Project "', p.Title, '" has reached 25% funding!')

FROM Donations d

JOIN Projects p ON d.ProjectID = p.ProjectID

WHERE d.ProjectID = NEW.ProjectID;

ELSEIF fundingPercentage >= 50 AND fundingPercentage < 75 THEN

INSERT INTO Notifications (UserID, ProjectID, Message)

SELECT DISTINCT d.UserID, NEW.ProjectID, CONCAT('Project "', p.Title, '" has reached 50% funding!')

FROM Donations d

JOIN Projects p ON d.ProjectID = p.ProjectID

WHERE d.ProjectID = NEW.ProjectID;

ELSEIF fundingPercentage >= 75 AND fundingPercentage < 100 THEN

INSERT INTO Notifications (UserID, ProjectID, Message)

SELECT DISTINCT d.UserID, NEW.ProjectID, CONCAT('Project "', p.Title, '" has reached 75% funding!')

FROM Donations d

JOIN Projects p ON d.ProjectID = p.ProjectID

WHERE d.ProjectID = NEW.ProjectID;

ELSEIF fundingPercentage >= 100 THEN

INSERT INTO Notifications (UserID, ProjectID, Message)

SELECT DISTINCT d.UserID, NEW.ProjectID, CONCAT('Project "', p.Title, '" has reached 100% funding and is now fully funded!')

FROM Donations d

JOIN Projects p ON d.ProjectID = p.ProjectID

WHERE d.ProjectID = NEW.ProjectID;

END IF;

END;

**-Write SQL queries to find top backers, most-funded projects, and reward distribution rates:**

### ****1. Query to Find Top Backers****

This query retrieves the top 5 backers based on the total amount of donations they made.

SELECT

u.UserID,

u.Name AS BackerName,

SUM(d.Amount) AS TotalDonations

FROM

Users u

JOIN

Donations d ON u.UserID = d.UserID

WHERE

u.UserType = 'Backer'

GROUP BY

u.UserID, u.Name

ORDER BY

TotalDonations DESC

LIMIT 5;

### ****2. Query to Find Most-Funded Projects****

This query lists the top 5 projects based on the total amount of funds raised.

SELECT

p.ProjectID,

p.Title AS ProjectTitle,

p.CreatorID,

u.Name AS CreatorName,

p.RaisedAmount,

p.GoalAmount,

(p.RaisedAmount / p.GoalAmount) \* 100 AS FundingPercentage

FROM

Projects p

JOIN

Users u ON p.CreatorID = u.UserID

ORDER BY

p.RaisedAmount DESC

LIMIT 5;

### ****3. Query to Calculate Reward Distribution Rates****

This query calculates the reward distribution rate for each project, showing the percentage of backers who have claimed rewards.

SELECT

p.ProjectID,

p.Title AS ProjectTitle,

COUNT(DISTINCT d.UserID) AS TotalBackers,

SUM(rt.BackersCount) AS RewardsClaimed,

(SUM(rt.BackersCount) / COUNT(DISTINCT d.UserID)) \* 100 AS RewardDistributionRate

FROM

Projects p

LEFT JOIN

Donations d ON p.ProjectID = d.ProjectID

LEFT JOIN

RewardTiers rt ON p.ProjectID = rt.ProjectID

GROUP BY

p.ProjectID, p.Title

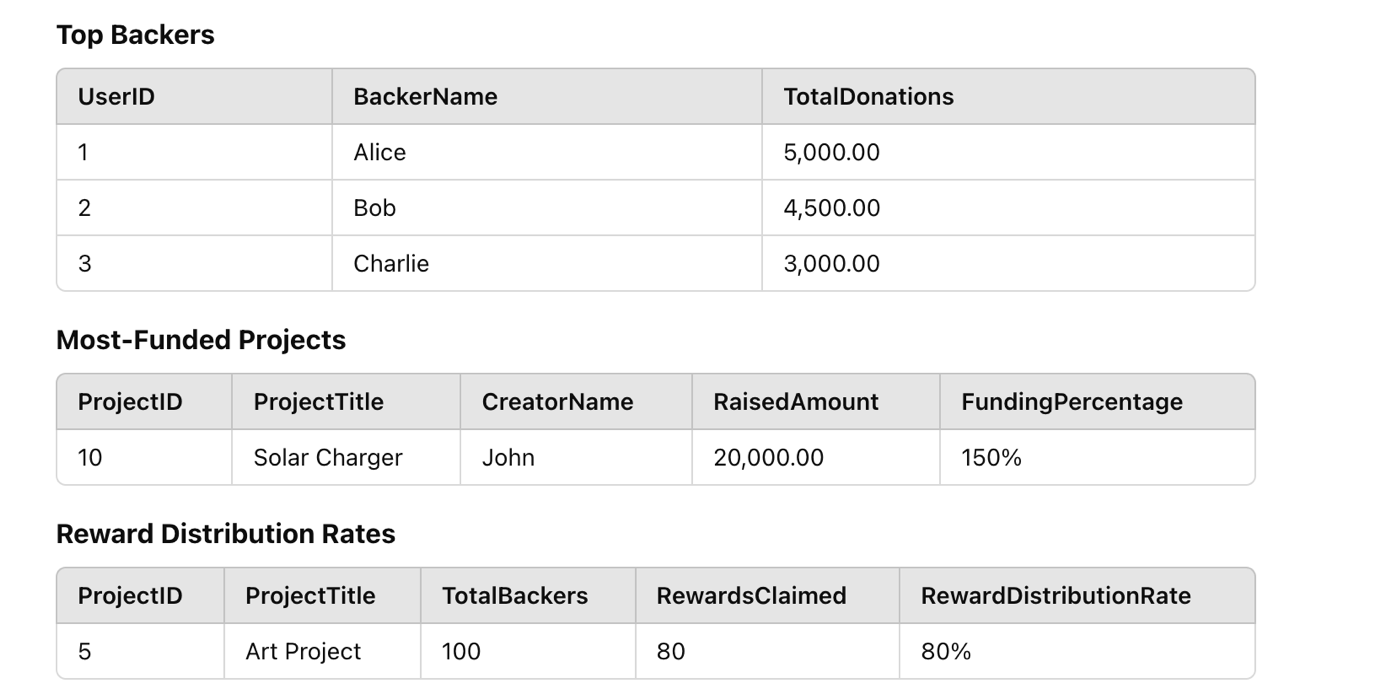
ORDER BY

RewardDistributionRate DESC;

### ****What Each Query Does****

1. **Top Backers**:
   * Aggregates donations by user and sorts them in descending order of total donations.
   * Limits the results to the top 5 backers.
2. **Most-Funded Projects**:
   * Orders projects by the total RaisedAmount.
   * Calculates the percentage of the funding goal achieved (FundingPercentage).
3. **Reward Distribution Rates**:
   * Counts the total backers (TotalBackers) and rewards claimed (RewardsClaimed) for each project.
   * Calculates the percentage of backers who have claimed rewards.

### ****Sample Outputs****



**CONCLUSION:**

The crowdfunding platform database is a well-structured system designed to manage projects, backers, donations, and rewards efficiently. With automated triggers to update project funding statuses and notify backers of milestones, the platform enhances transparency and engagement. Key insights from SQL queries, such as identifying top backers, most-funded projects, and reward distribution rates, enable data-driven decisions for administrators and project creators. The scalable schema supports seamless integration of advanced features like analytics, personalized notifications, and reward tracking, ensuring a user-friendly and impactful experience for both backers and creators, while laying a strong foundation for future enhancements.