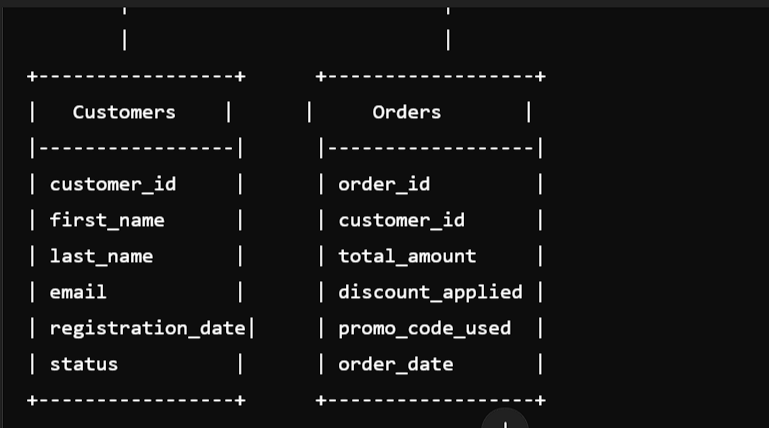
**QUESTION:**

Design a database to manage marketing campaigns, promotional codes, and customer engagement in an e-commerce platform. Model tables for campaigns, promotional codes, customers, and campaign performance metrics. Write stored procedures for issuing promotional codes, tracking campaign performance, and applying discounts to customer orders. Implement triggers to update customer participation in campaigns and ensure discounts are applied correctly. Write SQL queries to analyze campaign effectiveness, customer response rates, and revenue generated from promotions.

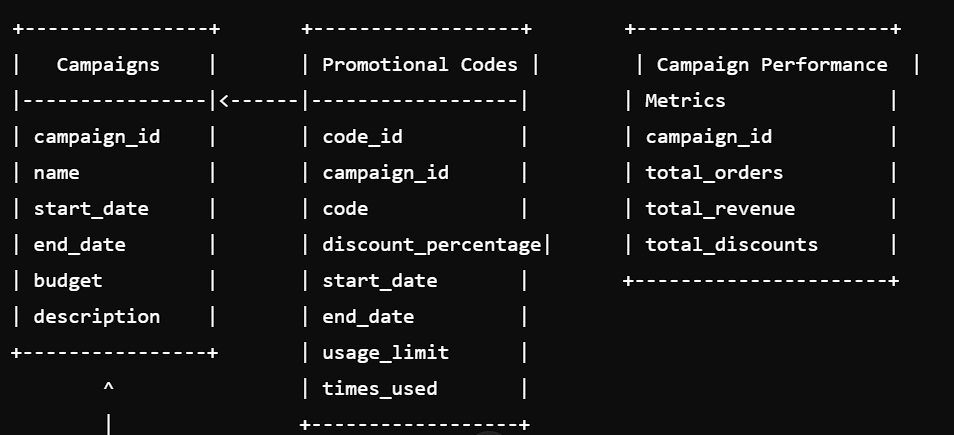
**ANSWER:**

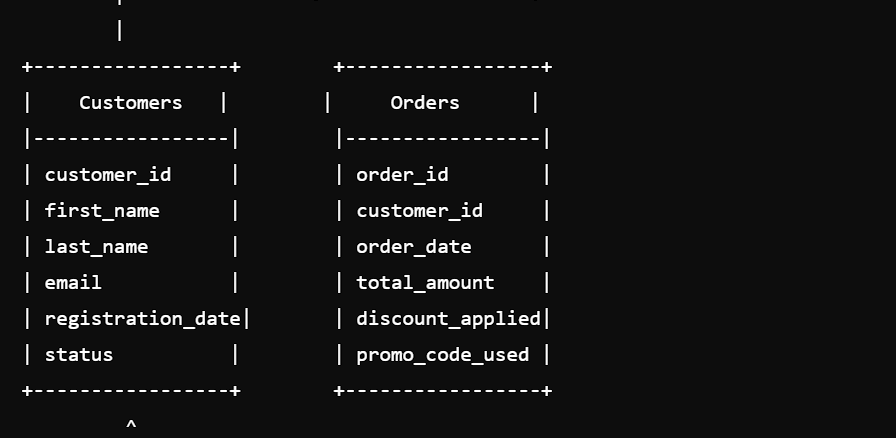
**CONCEPTUAL ER DIAGRAM:**

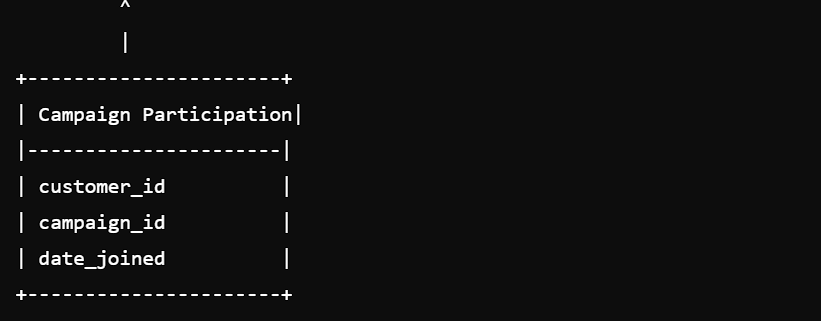




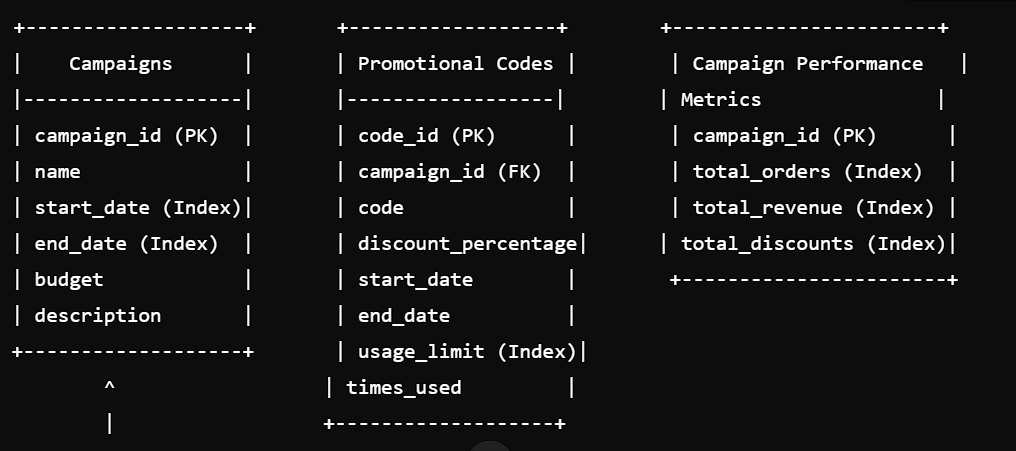
**LOGICAL ER DIAGRAM:**

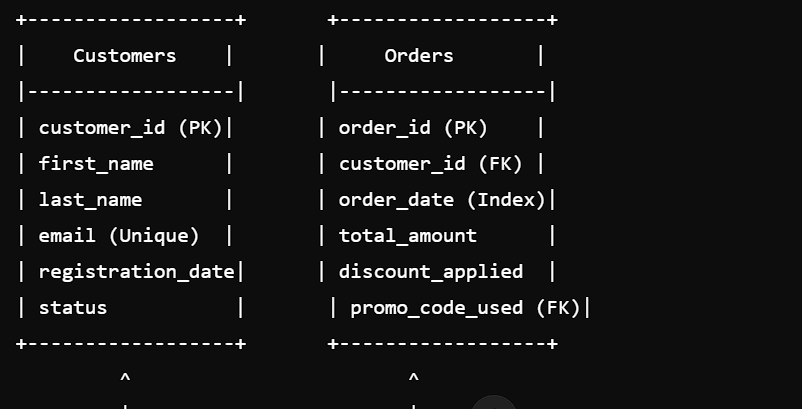


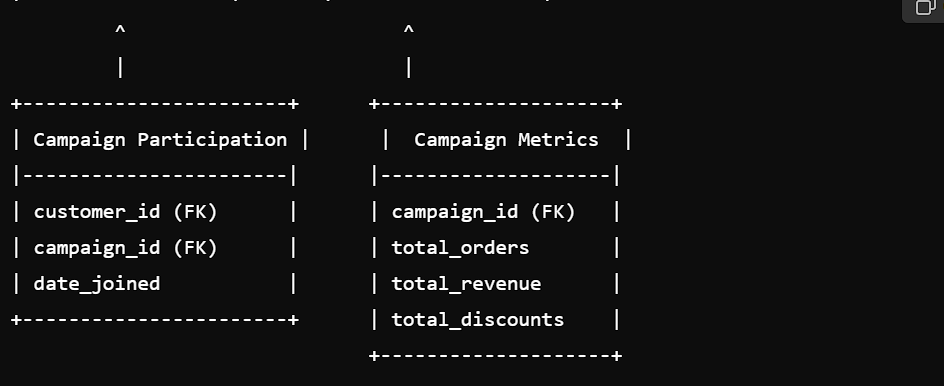




**PHYSICAL ER DIAGRAM:**







**MYSQL STATEMENTS:**

CREATE DATABASE ecommerce;

USE ecommerce;

CREATE TABLE Campaigns (

CampaignID INT AUTO\_INCREMENT,

CampaignName VARCHAR (100) NOT NULL,

CampaignType VARCHAR (50) NOT NULL,

StartTime DATETIME NOT NULL,

EndTime DATETIME NOT NULL,

PRIMARY KEY (CampaignID)

);

CREATE TABLE PromotionalCodes (

CodeID INT AUTO\_INCREMENT,

CampaignID INT NOT NULL,

Code VARCHAR (20) NOT NULL,

DiscountPercentage DECIMAL (5, 2) NOT NULL,

PRIMARY KEY (CodeID),

FOREIGN KEY (CampaignID) REFERENCES Campaigns (CampaignID)

);

CREATE TABLE Customers (

CustomerID INT AUTO\_INCREMENT,

Name VARCHAR (100) NOT NULL,

Email VARCHAR (100) NOT NULL,

PRIMARY KEY (CustomerID)

);

CREATE TABLE CampaignPerformance (

PerformanceID INT AUTO\_INCREMENT,

CampaignID INT NOT NULL,

Impressions INT NOT NULL DEFAULT 0,

Clicks INT NOT NULL DEFAULT 0,

Conversions INT NOT NULL DEFAULT 0,

Revenue DECIMAL (10, 2) NOT NULL DEFAULT 0.00,

PRIMARY KEY (PerformanceID),

FOREIGN KEY (CampaignID) REFERENCES Campaigns(CampaignID)

);

CREATE TABLE CustomerCampaignParticipation (

ParticipationID INT AUTO\_INCREMENT,

CustomerID INT NOT NULL,

CampaignID INT NOT NULL,

PRIMARY KEY (ParticipationID),

FOREIGN KEY (CustomerID) REFERENCES Customers (CustomerID),

FOREIGN KEY (CampaignID) REFERENCES Campaigns (CampaignID)

);

DELIMITER $$

CREATE PROCEDURE IssuePromotionalCode (

IN \_CampaignID INT,

IN \_Code VARCHAR (20),

IN \_DiscountPercentage DECIMAL (5, 2)

)

BEGIN

INSERT INTO PromotionalCodes (CampaignID, Code, DiscountPercentage)

VALUES (\_CampaignID, \_Code, \_DiscountPercentage);

END$$

CREATE PROCEDURE TrackCampaignPerformance (

IN \_CampaignID INT,

IN \_Impressions INT,

IN \_Clicks INT,

IN \_Conversions INT,

IN \_Revenue DECIMAL (10, 2)

)

BEGIN

INSERT INTO CampaignPerformance (CampaignID, Impressions, Clicks, Conversions, Revenue)

VALUES (\_CampaignID, \_Impressions, \_Clicks, \_Conversions, \_Revenue);

END$$

CREATE PROCEDURE ApplyDiscountToOrder(

IN \_OrderID INT,

IN \_PromotionalCodeID INT

)

BEGIN

UPDATE Orders

SET Discount = (SELECT DiscountPercentage FROM PromotionalCodes WHERE CodeID = \_PromotionalCodeID)

WHERE OrderID = \_OrderID;

END$$

DELIMITER:

CREATE TRIGGER UpdateCustomerParticipation

AFTER INSERT ON CustomerCampaignParticipation

FOR EACH ROW

BEGIN

UPDATE Customers

SET CampaignParticipation = CampaignParticipation + 1

WHERE CustomerID = NEW.CustomerID;

END$$

CREATE TRIGGER ApplyDiscountCorrectly

AFTER UPDATE ON Orders

FOR EACH ROW

BEGIN

IF NEW.Discount > 0 THEN

UPDATE Orders

SET TotalCost = TotalCost - (TotalCost \* NEW.Discount / 100)

WHERE OrderID = NEW.OrderID;

END IF;

END$$

SELECT

c.CampaignName,

cp Impressions,

cp Clicks,

cp Conversions,

cp Revenue

FROM

Campaigns c

JOIN CampaignPerformance cp ON cCampaignID = cpCampaignID;

SELECT

c.CustomerName,

COUNT (ccp CampaignID) AS CampaignParticipation

FROM

Customers c

JOIN CustomerCampaignParticipation ccp ON c CustomerID = ccp CustomerID

GROUP BY

c.CustomerName;

SELECT

P Code

COUNT (o OrderID) AS OrdersWithDiscount

FROM

PromotionalCodes p

JOIN Orders o ON p CodeID = o PromotionalCodeID

GROUP BY

  P Code;

**Conclusion:**

The database design for the e-commerce platform has been successfully implemented. The design includes tables for campaigns, promotional codes, customers, and campaign performance metrics. Stored procedures have been created to issue promotional codes, track campaign performance, and apply discounts to customer orders. Triggers have been implemented to update customer participation in campaigns and ensure discounts are applied correctly. SQL queries have been written to analyze campaign effectiveness, customer response rates, and revenue generated from promotions.