```
CREATE SCHEMA ccc_emergency_map;

-- Create and use the schema if it does not exist
CREATE DATABASE IF NOT EXISTS ccc_emergency_map;
USE ccc_emergency_map;

-- Drop tables if they exist to avoid conflicts
DROP TABLE IF EXISTS EmergencyIncidentAnalysis;
DROP TABLE IF EXISTS RouteConditionLog;
DROP TABLE IF EXISTS EmergencyResponseTeamMember;
DROP TABLE IF EXISTS EmergencyResponseTeam;
DROP TABLE IF EXISTS EmergencyResource;
DROP TABLE IF EXISTS EmergencyDistressAssignmentQueue;
DROP TABLE IF EXISTS CustomerFeedback;
DROP TABLE IF EXISTS EmergencyDistressAlerts;
DROP TABLE IF EXISTS CustomerTrip;
DROP TABLE IF EXISTS Trip;
DROP TABLE IF EXISTS TripType;
DROP TABLE IF EXISTS Geofences;
DROP TABLE IF EXISTS PointsOfInterest;
DROP TABLE IF EXISTS Route;
DROP TABLE IF EXISTS LocationsOfInterest;
DROP TABLE IF EXISTS Weather;
DROP TABLE IF EXISTS Employee;
DROP TABLE IF EXISTS Customer;
DROP TABLE IF EXISTS Account;
DROP TABLE IF EXISTS ActivityLog;
DROP TABLE IF EXISTS UserPermissions;
DROP TABLE IF EXISTS UserRoles;
DROP TABLE IF EXISTS RolePermissions;
DROP TABLE IF EXISTS Permissions;
DROP TABLE IF EXISTS Roles;

-- Account Table
CREATE TABLE Account (
    AccountID INT PRIMARY KEY AUTO_INCREMENT,
    Username VARCHAR(100) UNIQUE NOT NULL,
    PasswordHash VARBINARY(255) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    PhoneNumber VARCHAR(15),
    Address TEXT,
    RegistrationDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    AccountType ENUM('Customer', 'Employee') NOT NULL
);
```

```
-- Customer Table
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(100) NOT NULL,
    LastName VARCHAR(100) NOT NULL,
    DateOfBirth DATE,
    EmergencyContactName VARCHAR(100),
    EmergencyContactPhone VARCHAR(15),
    FOREIGN KEY (CustomerID) REFERENCES Account(AccountID)
);

-- Employee Table
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(100) NOT NULL,
    LastName VARCHAR(100) NOT NULL,
    Position VARCHAR(100) NOT NULL,
    HireDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Status VARCHAR(50) DEFAULT 'Available' NOT NULL,
    AccessLevel TINYINT DEFAULT 0,
    RoleUpdateRequired TINYINT DEFAULT 0,
    FOREIGN KEY (EmployeeID) REFERENCES Account(AccountID)
);

-- LocationsOfInterest Table
CREATE TABLE LocationsOfInterest (
    LocationID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Description TEXT,
    Latitude DECIMAL(10, 7) NOT NULL,
    Longitude DECIMAL(10, 7) NOT NULL,
    Type VARCHAR(50),
    Elevation DECIMAL(8, 2),
    TerrainType VARCHAR(50),
    AccessibilityNotes TEXT,
    LastUpdated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

```
-- Route Table
CREATE TABLE Route (
    RouteID INT PRIMARY KEY AUTO_INCREMENT,
    StartLocationID INT NOT NULL,
    EndLocationID INT NOT NULL,
    Distance DECIMAL(5, 2) NOT NULL,
    EstimatedTime DECIMAL(5, 2),
    DifficultyLevel VARCHAR(50),
    AgeRequirement INT,
    FOREIGN KEY (StartLocationID) REFERENCES LocationsOfInterest(LocationID),
    FOREIGN KEY (EndLocationID) REFERENCES LocationsOfInterest(LocationID)
);

-- PointsOfInterest Table
CREATE TABLE PointsOfInterest (
    POIID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Description TEXT,
    Latitude DECIMAL(10, 7) NOT NULL,
    Longitude DECIMAL(10, 7) NOT NULL,
    RouteID INT NOT NULL,
    Type VARCHAR(50),
    FOREIGN KEY (RouteID) REFERENCES Route(RouteID)
);

-- Geofences Table
CREATE TABLE Geofences (
    GeofenceID INT PRIMARY KEY AUTO_INCREMENT,
    POIID INT NOT NULL,
    Radius DECIMAL(6, 2) NOT NULL,
    AlertType VARCHAR(50),
    FOREIGN KEY (POIID) REFERENCES PointsOfInterest(POIID)
);

-- Weather Table
CREATE TABLE Weather (
    WeatherID INT PRIMARY KEY AUTO_INCREMENT,
    Description VARCHAR(100) NOT NULL,
    Temperature DECIMAL(5, 2),
    Humidity DECIMAL(5, 2),
    WindSpeed DECIMAL(5, 2),
    Cloudiness DECIMAL(5, 2), -- Optional, if needed for cloud cover specifics
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
-- TripType Table
CREATE TABLE TripType (
    TripTypeID INT PRIMARY KEY AUTO_INCREMENT,
    TypeName VARCHAR(100) UNIQUE NOT NULL,
    Description TEXT
);

-- Trip Table
CREATE TABLE Trip (
    TripID INT PRIMARY KEY AUTO_INCREMENT,
    RouteID INT NOT NULL,
    TripTypeID INT NOT NULL,
    WeatherID INT NOT NULL,
    TripDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    TotalDistance DECIMAL(5, 2),
    FOREIGN KEY (RouteID) REFERENCES Route(RouteID),
    FOREIGN KEY (TripTypeID) REFERENCES TripType(TripTypeID),
    FOREIGN KEY (WeatherID) REFERENCES Weather(WeatherID)
);

-- CustomerTrip Junction Table
CREATE TABLE CustomerTrip (
    CustomerTripID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT NOT NULL,
    TripID INT NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (TripID) REFERENCES Trip(TripID),
    UNIQUE(CustomerID, TripID)
);
```

```
-- EmergencyDistressAlerts Table
CREATE TABLE EmergencyDistressAlerts (
    AlertID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT NOT NULL,
    Latitude DECIMAL(10, 7) NOT NULL,
    Longitude DECIMAL(10, 7) NOT NULL,
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Status VARCHAR(50) NOT NULL,
    SeverityLevel ENUM('Low', 'Medium', 'High', 'Critical') NOT NULL DEFAULT 'Medium',
    ResponseTime DATETIME,
    ResolutionTime DATETIME,
    ResolutionNotes TEXT,
    AssignedEmployeeID INT,
    TripID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (AssignedEmployeeID) REFERENCES Employee(EmployeeID),
    FOREIGN KEY (TripID) REFERENCES Trip(TripID)
);

-- CustomerFeedback Table
CREATE TABLE CustomerFeedback (
    FeedbackID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT NOT NULL,
    TripID INT NOT NULL,
    TripTypeID INT NOT NULL,
    Feedback TEXT,
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (TripID) REFERENCES Trip(TripID),
    FOREIGN KEY (TripTypeID) REFERENCES TripType(TripTypeID)
);

-- EmergencyDistressAssignmentQueue Table
CREATE TABLE EmergencyDistressAssignmentQueue (
    QueueID INT PRIMARY KEY AUTO_INCREMENT,
    AlertID INT NOT NULL,
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Processed TINYINT(1) DEFAULT 0,
    FOREIGN KEY (AlertID) REFERENCES EmergencyDistressAlerts(AlertID)
);
```

```
-- EmergencyResource Table
CREATE TABLE EmergencyResource (
    ResourceID INT PRIMARY KEY AUTO_INCREMENT,
    ResourceName VARCHAR(100) NOT NULL,
    ResourceType ENUM('Vehicle', 'Equipment', 'Personnel') NOT NULL,
    CurrentLocationID INT,
    Status ENUM('Available', 'In Use', 'Out of Service') DEFAULT 'Available',
    LastUpdated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (CurrentLocationID) REFERENCES LocationsOfInterest(LocationID)
);

-- EmergencyResponseTeam Table
CREATE TABLE EmergencyResponseTeam (
    TeamID INT PRIMARY KEY AUTO_INCREMENT,
    TeamName VARCHAR(100) NOT NULL,
    LeadEmployeeID INT,
    FOREIGN KEY (LeadEmployeeID) REFERENCES Employee(EmployeeID)
);

-- EmergencyResponseTeamMember Junction Table
CREATE TABLE EmergencyResponseTeamMember (
    TeamID INT,
    EmployeeID INT,
    Role VARCHAR(50),
    PRIMARY KEY (TeamID, EmployeeID),
    FOREIGN KEY (TeamID) REFERENCES EmergencyResponseTeam(TeamID),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

-- RouteConditionLog Table
CREATE TABLE RouteConditionLog (
    LogID INT PRIMARY KEY AUTO_INCREMENT,
    RouteID INT NOT NULL,
    ConditionType ENUM('Normal', 'Caution', 'Danger', 'Closed') NOT NULL,
    Description TEXT,
    ReportedBy INT,
    ReportedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ResolvedAt TIMESTAMP NULL,
    FOREIGN KEY (RouteID) REFERENCES Route(RouteID),
    FOREIGN KEY (ReportedBy) REFERENCES Employee(EmployeeID)
);
```

```
-- EmergencyIncidentAnalysis Table
CREATE TABLE EmergencyIncidentAnalysis (
    AnalysisID INT PRIMARY KEY AUTO_INCREMENT,
    AlertID INT NOT NULL,
    ResponseTime INT,  -- in minutes
    ResolutionTime INT,  -- in minutes
    SuccessFactors TEXT,
    ImprovementAreas TEXT,
    AnalyzedBy INT,
    AnalysisDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (AlertID) REFERENCES EmergencyDistressAlerts(AlertID),
    FOREIGN KEY (AnalyzedBy) REFERENCES Employee(EmployeeID)
);

-- Redefine the delimiter
DELIMITER //

-- Trigger to insert activity log on changes to Account table
CREATE TRIGGER trg_Account_AfterInsert
AFTER INSERT ON Account
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Account Created', 'Account', NEW.AccountID, 'INSERT', CONCAT('Account
created with Username: ', NEW.Username), 'SYSTEM');
END;
//

CREATE TRIGGER trg_Account_AfterUpdate
AFTER UPDATE ON Account
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Account Updated', 'Account', NEW.AccountID, 'UPDATE', CONCAT('Account
updated with Username: ', NEW.Username), 'SYSTEM');
END;
//
```

```
CREATE TRIGGER trg_Account_AfterDelete
AFTER DELETE ON Account
FOR EACH ROW
BEGIN
   INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
   VALUES ('Account Deleted', 'Account', OLD.AccountID, 'DELETE', CONCAT('Account
deleted with Username: ', OLD.Username), 'SYSTEM');
END;
//

-- Trigger to insert activity log on changes to Customer table
CREATE TRIGGER trg_Customer_AfterInsert
AFTER INSERT ON Customer
FOR EACH ROW
BEGIN
   INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
   VALUES ('Customer Created', 'Customer', NEW.CustomerID, 'INSERT', CONCAT('Customer
created with Name: ', NEW.FirstName, ' ', NEW.LastName), 'SYSTEM');
END;
//

CREATE TRIGGER trg_Customer_AfterUpdate
AFTER UPDATE ON Customer
FOR EACH ROW
BEGIN
   INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
   VALUES ('Customer Updated', 'Customer', NEW.CustomerID, 'UPDATE', CONCAT('Customer
updated with Name: ', NEW.FirstName, ' ', NEW.LastName), 'SYSTEM');
END;
//

CREATE TRIGGER trg_Customer_AfterDelete
AFTER DELETE ON Customer
FOR EACH ROW
BEGIN
   INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
   VALUES ('Customer Deleted', 'Customer', OLD.CustomerID, 'DELETE', CONCAT('Customer
deleted with Name: ', OLD.FirstName, ' ', OLD.LastName), 'SYSTEM');
END;
```

```
//

-- Trigger to insert activity log on changes to Employee table
CREATE TRIGGER trg_Employee_AfterInsert
AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Employee Created', 'Employee', NEW.EmployeeID, 'INSERT', CONCAT('Employee
created with Name: ', NEW.FirstName, ' ', NEW.LastName), 'SYSTEM');
END;
//

CREATE TRIGGER trg_Employee_AfterUpdate
AFTER UPDATE ON Employee
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Employee Updated', 'Employee', NEW.EmployeeID, 'UPDATE',
CONCAT('Employee updated with Name: ', NEW.FirstName, ' ', NEW.LastName), 'SYSTEM');
END;
//

CREATE TRIGGER trg_Employee_AfterDelete
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Employee Deleted', 'Employee', OLD.EmployeeID, 'DELETE', CONCAT('Employee
deleted with Name: ', OLD.FirstName, ' ', OLD.LastName), 'SYSTEM');
END;
//
```

```
-- Trigger to insert activity log on changes to Trip table
CREATE TRIGGER trg_Trip_AfterInsert
AFTER INSERT ON Trip
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Trip Created', 'Trip', NEW.TripID, 'INSERT', CONCAT('Trip created with RouteID: ',
NEW.RouteID, ' and TripTypeID: ', NEW.TripTypeID), 'SYSTEM');
END;
//

CREATE TRIGGER trg_Trip_AfterUpdate
AFTER UPDATE ON Trip
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Trip Updated', 'Trip', NEW.TripID, 'UPDATE', CONCAT('Trip updated with RouteID:
', NEW.RouteID, ' and TripTypeID: ', NEW.TripTypeID), 'SYSTEM');
END;
//

CREATE TRIGGER trg_Trip_AfterDelete
AFTER DELETE ON Trip
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Trip Deleted', 'Trip', OLD.TripID, 'DELETE', CONCAT('Trip deleted with RouteID: ',
OLD.RouteID, ' and TripTypeID: ', OLD.TripTypeID), 'SYSTEM');
END;
//
```

```
-- Trigger to insert activity log on changes to EmergencyDistressAlerts table
CREATE TRIGGER trg_EmergencyDistressAlerts_AfterInsert
AFTER INSERT ON EmergencyDistressAlerts
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Distress Alert Created', 'EmergencyDistressAlerts', NEW.AlertID,
'INSERT', CONCAT('Alert created for CustomerID: ', NEW.CustomerID, ' with SeverityLevel: ',
NEW.SeverityLevel), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyDistressAlerts_AfterUpdate
AFTER UPDATE ON EmergencyDistressAlerts
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Distress Alert Updated', 'EmergencyDistressAlerts', NEW.AlertID,
'UPDATE', CONCAT('Alert updated for CustomerID: ', NEW.CustomerID, ' with SeverityLevel: ',
NEW.SeverityLevel), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyDistressAlerts_AfterDelete
AFTER DELETE ON EmergencyDistressAlerts
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Distress Alert Deleted', 'EmergencyDistressAlerts', OLD.AlertID,
'DELETE', CONCAT('Alert deleted for CustomerID: ', OLD.CustomerID, ' with SeverityLevel: ',
OLD.SeverityLevel), 'SYSTEM');
END;
//
```

```
-- Trigger to insert activity log on changes to CustomerFeedback table
CREATE TRIGGER trg_CustomerFeedback_AfterInsert
AFTER INSERT ON CustomerFeedback
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Customer Feedback Created', 'CustomerFeedback', NEW.FeedbackID, 'INSERT',
CONCAT('Feedback created for TripID: ', NEW.TripID, ' with Rating: ', NEW.Rating), 'SYSTEM');
END;
//

CREATE TRIGGER trg_CustomerFeedback_AfterUpdate
AFTER UPDATE ON CustomerFeedback
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Customer Feedback Updated', 'CustomerFeedback', NEW.FeedbackID, 'UPDATE',
CONCAT('Feedback updated for TripID: ', NEW.TripID, ' with Rating: ', NEW.Rating),
'SYSTEM');
END;
//

CREATE TRIGGER trg_CustomerFeedback_AfterDelete
AFTER DELETE ON CustomerFeedback
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Customer Feedback Deleted', 'CustomerFeedback', OLD.FeedbackID, 'DELETE',
CONCAT('Feedback deleted for TripID: ', OLD.TripID, ' with Rating: ', OLD.Rating), 'SYSTEM');
END;
//
```

```
-- Trigger to insert activity log on changes to EmergencyResource table
CREATE TRIGGER trg_EmergencyResource_AfterInsert
AFTER INSERT ON EmergencyResource
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Resource Created', 'EmergencyResource', NEW.ResourceID,
'INSERT', CONCAT('Resource created with Name: ', NEW.ResourceName, ' and Type: ',
NEW.ResourceType), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyResource_AfterUpdate
AFTER UPDATE ON EmergencyResource
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Resource Updated', 'EmergencyResource', NEW.ResourceID,
'UPDATE', CONCAT('Resource updated with Name: ', NEW.ResourceName, ' and Type: ',
NEW.ResourceType), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyResource_AfterDelete
AFTER DELETE ON EmergencyResource
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Resource Deleted', 'EmergencyResource', OLD.ResourceID,
'DELETE', CONCAT('Resource deleted with Name: ', OLD.ResourceName, ' and Type: ',
OLD.ResourceType), 'SYSTEM');
END;
//
```

```
-- Trigger to insert activity log on changes to EmergencyResponseTeam table
CREATE TRIGGER trg_EmergencyResponseTeam_AfterInsert
AFTER INSERT ON EmergencyResponseTeam
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Response Team Created', 'EmergencyResponseTeam', NEW.TeamID,
'INSERT', CONCAT('Team created with Name: ', NEW.TeamName), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyResponseTeam_AfterUpdate
AFTER UPDATE ON EmergencyResponseTeam
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Response Team Updated', 'EmergencyResponseTeam', NEW.TeamID,
'UPDATE', CONCAT('Team updated with Name: ', NEW.TeamName), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyResponseTeam_AfterDelete
AFTER DELETE ON EmergencyResponseTeam
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Response Team Deleted', 'EmergencyResponseTeam', OLD.TeamID,
'DELETE', CONCAT('Team deleted with Name: ', OLD.TeamName), 'SYSTEM');
END;
//
```

```
-- Trigger to insert activity log on changes to EmergencyDistressAssignmentQueue table
CREATE TRIGGER trg_EmergencyDistressAssignmentQueue_AfterInsert
AFTER INSERT ON EmergencyDistressAssignmentQueue
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Distress Assignment Queue Created',
'EmergencyDistressAssignmentQueue', NEW.QueueID, 'INSERT', CONCAT('Queue created
with AlertID: ', NEW.AlertID), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyDistressAssignmentQueue_AfterUpdate
AFTER UPDATE ON EmergencyDistressAssignmentQueue
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Distress Assignment Queue Updated',
'EmergencyDistressAssignmentQueue', NEW.QueueID, 'UPDATE', CONCAT('Queue updated
with AlertID: ', NEW.AlertID), 'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyDistressAssignmentQueue_AfterDelete
AFTER DELETE ON EmergencyDistressAssignmentQueue
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Distress Assignment Queue Deleted',
'EmergencyDistressAssignmentQueue', OLD.QueueID, 'DELETE', CONCAT('Queue deleted
with AlertID: ', OLD.AlertID), 'SYSTEM');
END;
//

DELIMITER //

-- Trigger to handle merging customers into existing trips
CREATE TRIGGER trg_MergeCustomerIntoTrip
AFTER INSERT ON CustomerTrip
FOR EACH ROW
```

```
BEGIN
    -- Check if the TripID already exists in the CustomerTrip table
    IF EXISTS (SELECT 1 FROM CustomerTrip WHERE TripID = NEW.TripID) THEN
        -- TripID exists, ensure the new customer is added to the existing trip
        -- Insert the new customer into the junction table if not already present
        INSERT IGNORE INTO CustomerTrip (CustomerID, TripID)
        VALUES (NEW.CustomerID, NEW.TripID);
    END IF;
END //
-- Reset DELIMITER
DELIMITER ;
```