```
-- 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 Route;
DROP TABLE IF EXISTS LocationsOfInterest;
DROP TABLE IF EXISTS Permissions;
DROP TABLE IF EXISTS Account;
DROP TABLE IF EXISTS Customer;
DROP TABLE IF EXISTS Employee;
DROP TABLE IF EXISTS ActivityLog;

-- Permissions Table
CREATE TABLE Permissions (
    PermissionID TINYINT PRIMARY KEY,
    PermissionName VARCHAR(100) UNIQUE NOT NULL
);

-- Set default permissions
INSERT INTO Permissions (PermissionID, PermissionName) VALUES
(1, 'Customer'),
(2, 'Employee'),
(3, 'Admin'),
(4, 'Super Admin');

-- 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,
    FirstName VARCHAR(100) NOT NULL,
    LastName VARCHAR(100) NOT NULL,
    AccountType ENUM('Customer', 'Employee') NOT NULL,
    PermissionTier TINYINT NOT NULL DEFAULT 0, -- Default to no access
    FOREIGN KEY (PermissionTier) REFERENCES Permissions(PermissionID)
);

-- Customer Table
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    EmergencyContactName VARCHAR(100),
    EmergencyContactPhone VARCHAR(15),
    FOREIGN KEY (CustomerID) REFERENCES Account(AccountID)
);

-- Employee Table
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    Position VARCHAR(100) NOT NULL,
    HireDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Status VARCHAR(50) DEFAULT 'Available' NOT NULL,
    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)
);

-- 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,
```

```
    TripDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    TotalDistance DECIMAL(5, 2),
    FOREIGN KEY (RouteID) REFERENCES Route(RouteID),
    FOREIGN KEY (TripTypeID) REFERENCES TripType(TripTypeID)
);

-- 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,
    CustomerTripID 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,
    FOREIGN KEY (CustomerTripID) REFERENCES CustomerTrip(CustomerTripID),
    FOREIGN KEY (AssignedEmployeeID) REFERENCES Employee(EmployeeID)
);

-- CustomerFeedback Table
CREATE TABLE CustomerFeedback (
    FeedbackID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerTripID INT NOT NULL,
    Feedback TEXT,
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (CustomerTripID) REFERENCES CustomerTrip(CustomerTripID)
);

-- 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)
);

-- ActivityLog Table
CREATE TABLE ActivityLog (
    LogID INT PRIMARY KEY AUTO_INCREMENT,
    ActivityType VARCHAR(100),
    TableName VARCHAR(100),
    RecordID INT,
    OperationType VARCHAR(10),
    Description TEXT,
    Username VARCHAR(100),
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Define triggers for activity logging
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', OLD.AccountID, 'UPDATE', CONCAT('Account
updated with Username: ', OLD.Username), 'SYSTEM');
END;
//

DELIMITER ;
DELIMITER //

-- 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 ID: ', NEW.CustomerID), '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', OLD.CustomerID, 'UPDATE', CONCAT('Customer
updated with ID: ', OLD.CustomerID), '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 ID: ', NEW.EmployeeID), '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', OLD.EmployeeID, 'UPDATE',
CONCAT('Employee updated with ID: ', OLD.EmployeeID), '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 ID: ',
NEW.TripID), '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', OLD.TripID, 'UPDATE', CONCAT('Trip updated with ID: ',
OLD.TripID), '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('Emergency distress alert created with ID: ', NEW.AlertID), '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', OLD.AlertID,
'UPDATE', CONCAT('Emergency distress alert updated with ID: ', OLD.AlertID), '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('Customer feedback created with ID: ', NEW.FeedbackID), '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', OLD.FeedbackID, 'UPDATE',
CONCAT('Customer feedback updated with ID: ', OLD.FeedbackID), '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('Emergency response team created with ID: ', NEW.TeamID), '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', OLD.TeamID,
'UPDATE', CONCAT('Emergency response team updated with ID: ', OLD.TeamID), 'SYSTEM');
END;
//

-- Trigger to insert activity log on changes to EmergencyResponseTeamMember table
CREATE TRIGGER trg_EmergencyResponseTeamMember_AfterInsert
AFTER INSERT ON EmergencyResponseTeamMember
FOR EACH ROW
BEGIN
   INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
   VALUES ('Emergency Response Team Member Added', 'EmergencyResponseTeamMember',
CONCAT(NEW.TeamID, '-', NEW.EmployeeID), 'INSERT', CONCAT('Emergency response team
member added to TeamID: ', NEW.TeamID, ' with EmployeeID: ', NEW.EmployeeID),
'SYSTEM');
END;
//

CREATE TRIGGER trg_EmergencyResponseTeamMember_AfterUpdate
AFTER UPDATE ON EmergencyResponseTeamMember
FOR EACH ROW
BEGIN
```

```
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Response Team Member Updated',
'EmergencyResponseTeamMember', CONCAT(OLD.TeamID, '-', OLD.EmployeeID), 'UPDATE',
CONCAT('Emergency response team member updated for TeamID: ', OLD.TeamID, ' with
EmployeeID: ', OLD.EmployeeID), 'SYSTEM');
END;
//

-- Trigger to insert activity log on changes to RouteConditionLog table
CREATE TRIGGER trg_RouteConditionLog_AfterInsert
AFTER INSERT ON RouteConditionLog
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Route Condition Log Created', 'RouteConditionLog', NEW.LogID, 'INSERT',
CONCAT('Route condition log created with ID: ', NEW.LogID), 'SYSTEM');
END;
//

CREATE TRIGGER trg_RouteConditionLog_AfterUpdate
AFTER UPDATE ON RouteConditionLog
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Route Condition Log Updated', 'RouteConditionLog', OLD.LogID, 'UPDATE',
CONCAT('Route condition log updated with ID: ', OLD.LogID), 'SYSTEM');
END;
//

-- Trigger to insert activity log on changes to EmergencyIncidentAnalysis table
CREATE TRIGGER trg_EmergencyIncidentAnalysis_AfterInsert
AFTER INSERT ON EmergencyIncidentAnalysis
FOR EACH ROW
BEGIN
    INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
    VALUES ('Emergency Incident Analysis Created', 'EmergencyIncidentAnalysis',
NEW.AnalysisID, 'INSERT', CONCAT('Emergency incident analysis created with ID: ',
NEW.AnalysisID), 'SYSTEM');
END;
//
```

```
CREATE TRIGGER trg_EmergencyIncidentAnalysis_AfterUpdate
AFTER UPDATE ON EmergencyIncidentAnalysis
FOR EACH ROW
BEGIN
   INSERT INTO ActivityLog (ActivityType, TableName, RecordID, OperationType, Description,
Username)
   VALUES ('Emergency Incident Analysis Updated', 'EmergencyIncidentAnalysis',
OLD.AnalysisID, 'UPDATE', CONCAT('Emergency incident analysis updated with ID: ',
OLD.AnalysisID), '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('Emergency resource created with ID: ', NEW.ResourceID), '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', OLD.ResourceID,
'UPDATE', CONCAT('Emergency resource updated with ID: ', OLD.ResourceID), 'SYSTEM');
END;
//

DELIMITER ;

DELIMITER $$

-- Trigger to log new account creation
CREATE TRIGGER trg_after_account_insert
AFTER INSERT ON Account
FOR EACH ROW
```

```
BEGIN
    INSERT INTO ActivityLog (AccountID, ActionType, ActionDetails, ActionTimestamp)
    VALUES (NEW.AccountID, 'Account Created', CONCAT('Account created for user ',
NEW.Username), NOW());
END$$

-- Trigger to log trip request and ensure all members of the trip are added to TripJunction table
CREATE TRIGGER trg_after_trip_request_insert
AFTER INSERT ON TripRequest
FOR EACH ROW
BEGIN
    DECLARE existing_trip_id INT;

    -- Check if the requested trip exists
    SELECT TripID INTO existing_trip_id
    FROM Trip
    WHERE Destination = NEW.Destination
      AND StartDate = NEW.StartDate
      AND EndDate = NEW.EndDate;

    -- If the trip exists, add all members of the trip to the TripJunction table
    IF existing_trip_id IS NOT NULL THEN
        INSERT INTO TripJunction (TripID, CustomerID)
        SELECT existing_trip_id, CustomerID
        FROM TripRequest
        WHERE TripRequest.TripID = existing_trip_id;
    ELSE
        -- If the trip doesn't exist, create a new trip and add the current customer to TripJunction
        INSERT INTO Trip (Destination, StartDate, EndDate)
        VALUES (NEW.Destination, NEW.StartDate, NEW.EndDate);

        SET existing_trip_id = LAST_INSERT_ID();

        INSERT INTO TripJunction (TripID, CustomerID)
        VALUES (existing_trip_id, NEW.CustomerID);
    END IF;
END$$

-- Trigger to delete logs older than 30 days
CREATE TRIGGER trg_after_log_cleanup
AFTER INSERT ON ActivityLog
FOR EACH ROW
BEGIN
    DELETE FROM ActivityLog
```

```
    WHERE ActionTimestamp < NOW() - INTERVAL 30 DAY;
END$$

DELIMITER ;
```