

**SCHOOL OF BUSINESS AND ECONOMICS**

**DEPARTMENT OF BUSINESS TECHNOLOGY**

**DATABASE MANAGEMENT**

**Project Name:**

**AMBULANCE BOOKING SYSTEM**

**Student name: HABIYAREMYE LAURIER**

**Reg No: 222003068**

**Submitted to: Dr. BUGINGO Emmanuel**

**Date: 14/ 09/ 2023**

**SECTION 1**

**Q1. . Describe all the entities and their corresponding attributes that are in your database.**

**User :** A User Entity will contain all information that concerns to patients that created an account in the system. There attributes will be as follow:

***UserID***: A unique identifier for each user.

***Name:*** The name of the user.

***PhoneNumber:*** The phone number of the user.

***Email:*** The email address of the user.

***Address:*** The address of the user.

***DateOfBirth:*** The date of birth of the user.

***Gender:*** The gender of the user.

***MedicalHistory:*** Information about the user's medical history.

***InsuranceInfo:*** Information about the user's insurance .

**AMBULANCE Entity:** An ambulance entity will hold all information about Ambulances in the system and it will contain information about the ambulance drivers. The attributes will be as follows:

***AmbulanceID:*** A unique identifier for each ambulance.

***VehicleNumber:*** The number or plate number of the ambulance.

***Type:*** The type or category of the ambulance.

***Capacity:*** The maximum number of passengers the ambulance can carry.

***Equipment:*** Information about the equipment available in the ambulance .

***CurrentLatitude:*** The current latitude coordinate of the ambulance's location.

***CurrentLongitude:*** The current longitude coordinate of the ambulance's location.

***AvailabilityStatus:*** The status indicating whether the ambulance is available.

***DriverName:*** The name of the ambulance driver.

***DriverContact:*** The contact information of the ambulance driver.

***DriverCertification:*** Certification details of the ambulance driver.

**BOOKING REQUEST Entity:** This entity will contain all information about the bookings that has been made by the users, it will contain information about the Hospitals that have been chosen as the destination of the patient. The attributes will be as follow:

***BookingID:*** A unique identifier for each booking request.

***UserID:*** The ID of the user making the booking (foreign key referencing User table).

***PickupLatitude:*** The latitude coordinate for the pickup location.

***PickupLongitude:*** The longitude coordinate for the pickup location.

***DestinationLatitude:*** The latitude coordinate for the destination.

***DestinationLongitude:*** The longitude coordinate for the destination.

***HospitalName:*** The name of the hospital (if applicable).

***RequestedTime:*** The date and time when the booking was requested.

***PriorityLevel:*** The priority level of the booking.

***Status:*** The status of the booking.

***Reason:*** The reason for the booking .

***AdditionalNotes:*** Additional notes or comments related to the booking .

**TRIP ENTITY**: This entity will hold all information that concerns to the Travels and trips done according to the bookings that were made. The attributes will be as follow;

***TripID:*** A unique identifier for each trip.

***BookingID:*** The ID of the booking associated with the trip (foreign key referencing BookingRequest table).

***AmbulanceID:*** The ID of the ambulance assigned to the trip (foreign key referencing Ambulance table).

***PickupTime:*** The date and time of the trip's pickup.

***DropoffTime:*** The date and time of the trip's drop-off.

***DistanceTraveled:*** The distance traveled during the trip.

***EstimatedArrivalTime:*** The estimated arrival time at the destination.

***Fare:*** The fare or cost of the trip.

***PaymentStatus:*** The payment status of the trip.

**PAYMENT ENTITY:** This Entity will contain information about the payments that will be done according to the trips taken by the users, it show whether the trip was paid or not. The attributes will be as follow;

***PaymentID:*** A unique identifier for each payment.

***TripID:*** The ID of the trip associated with the payment (foreign key referencing Trip table).

***UserID:*** The ID of the user making the payment (foreign key referencing User table).

***PaymentMethod***: The method used for payment.

***Amount:*** The amount paid.

***PaymentDateTime:*** The date and time of the payment.

**EMERGENCY CONTACTS ENTITY:** This entity will hold information about the emergency contacts of the patients signed up in the system, These contacts will be used incase there is any issue that needs to be told to the family of the patient. The attributes will be as follow;

***ContactID***: A unique identifier for each emergency contact.

***UserID***: The ID of the user associated with the emergency contact (foreign key referencing User table).

***ContactName:*** The name of the emergency contact.

***Relationship:*** The relationship between the user and the emergency contact.

***ContactPhoneNumber:*** The phone number of the emergency contact.

**FEEDBACK AND RATINGS Entity:** This entity will show and contain all feedbacks of the users and it will help to see the ratings of our users about the service we delivered to them. The attributes will be as follow;

***FeedbackID:*** A unique identifier for each feedback and rating.

***TripID:*** The ID of the trip associated with the feedback and rating (foreign key referencing Trip table).

***UserID:*** The ID of the user providing feedback and rating (foreign key referencing User table).

***Rating:*** The rating provided by the user.

***Comments:*** Comments or feedback from the user.

***FeedbackDateTime:*** The date and time when the feedback was provided.

**ADMIN ENTITY:** This will contain all information about the admins in the system, this will differentiate a Dispatcher from Coordinator or an administrator in the system. The attributes will be as follow;

***AdminID:*** A unique identifier for each admin.

***Name:*** The name of the admin.

***PhoneNumber:*** The phone number of the admin.

***Email:*** The email address of the admin.

***Role:*** The role or position of the admin.

**NOTIFICATION ENTITY:** This entity will show and contain the notifications about the actions in the system, this will show if a user has received a notification, this will help us to see if there is any new user created, if an ambulance is available or any action done in the system. The attributes will be as follow;

***NotificationID:*** A unique identifier for each notification.

***UserID:*** The ID of the user associated with the notification (foreign key referencing User table).

***Content:*** The content of the notification .

***Timestamp:*** The timestamp indicating when the notification was sent.

***Status***: The status of the notification.

**REPORTS AND ANALYTICS ENTITY:** This entity will contain the report, analytics and views of the administrators as there are the coordinators and auditors in the system. The attributes will be as follow;

***ReportID:*** A unique identifier for each report.

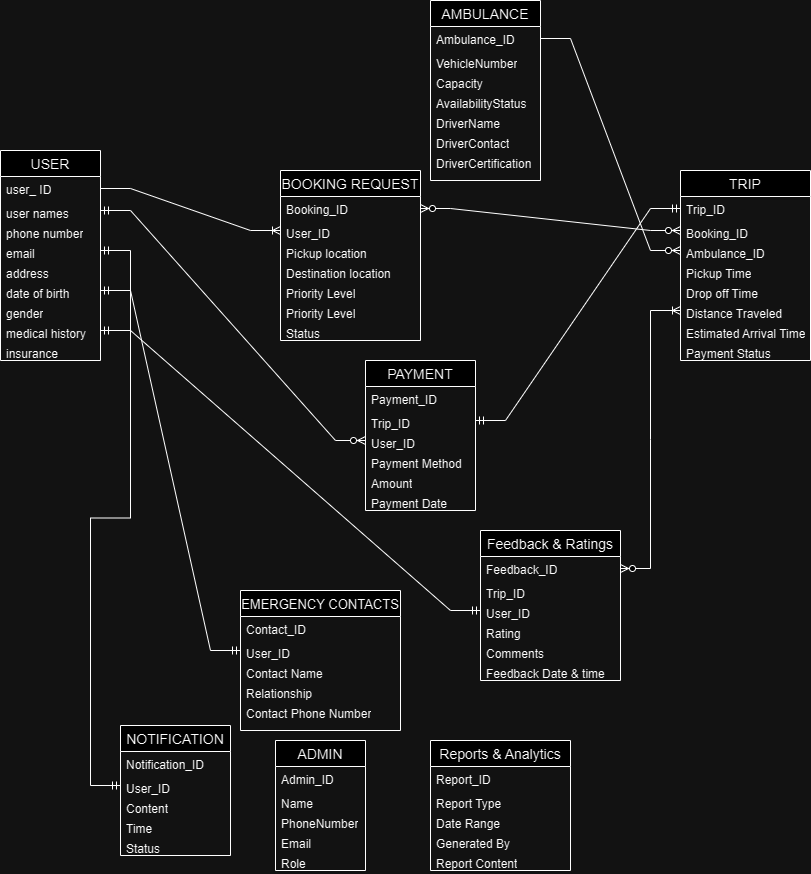
***ReportType:*** The type or category of the report.

***DateRange:*** The date range covered by the report.

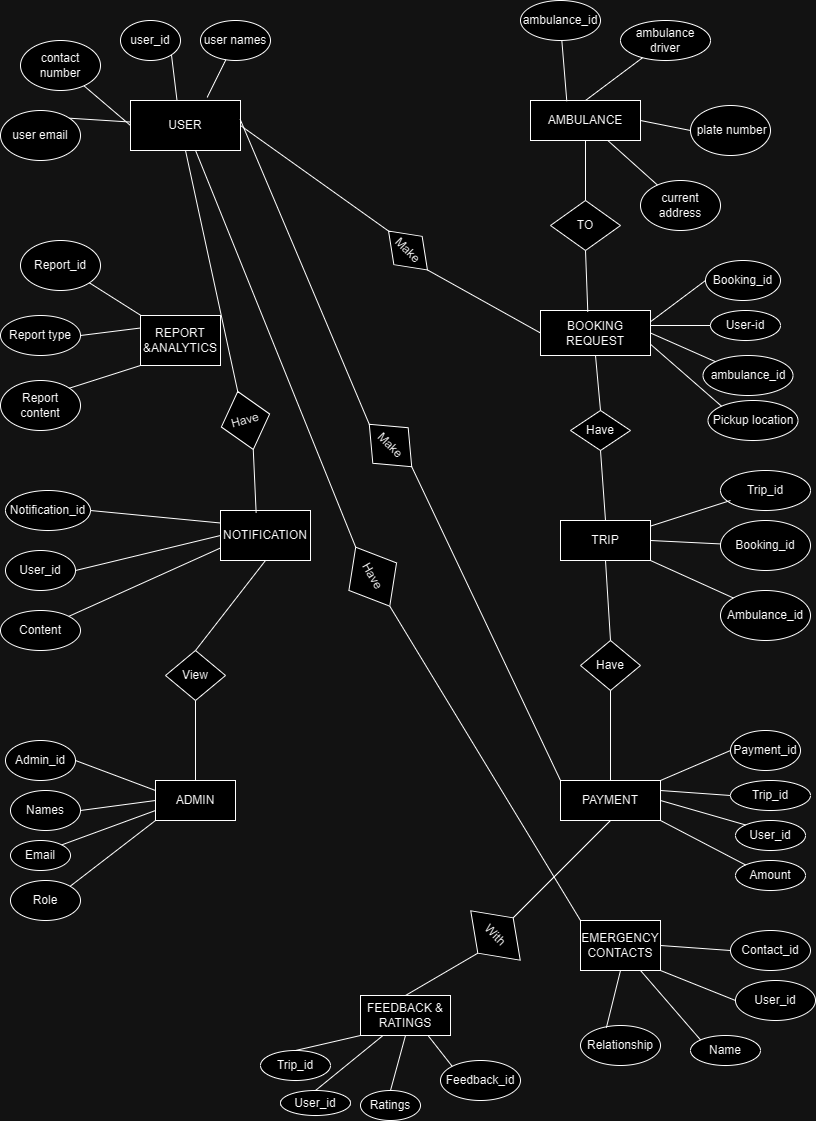
***GeneratedBy:*** The ID of the admin who generated the report (foreign key referencing Admin table).

***ReportContent:*** The content or details of the report .

**Q2. Create an LDM of your entities**



Q3. Create an ERD



**SECTION II SQL**

Q1. Create the database of your system

CREATE DATABASE Ambulance\_booking\_syst;

Q2. Write queries to create all the tables and relationships of your system

**1.USER TABLE**

CREATE TABLE User (

UserID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(255),

PhoneNumber VARCHAR(20),

Email VARCHAR(255),

Address VARCHAR(255),

DateOfBirth DATE,

Gender VARCHAR(10),

Password VARCHAR(255),

InsuranceTEXT

);

**2.AMBULANCE TABLE**

CREATE TABLE Ambulance (

AmbulanceID INT AUTO\_INCREMENT PRIMARY KEY,

VehicleNumber VARCHAR(20),

Type VARCHAR(50),

Capacity INT,

Equipment TEXT,

CurrentLocation VARCHAR(50),

DriverName VARCHAR(255),

DriverContact VARCHAR(20),

DriverCertification VARCHAR(255)

);

**3.BOOKING REQUEST TABLE**

CREATE TABLE BookingRequest (

BookingID INT AUTO\_INCREMENT PRIMARY KEY,

UserID INT,

PickupLocation VARCHAR(50),

Destination VARCHAR(50),

HospitalName VARCHAR(50),

RequestedTime DATETIME,

PriorityLevel VARCHAR(20),

Reason TEXT,

AdditionalNotes TEXT,

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**4.TRIP TABLE**

CREATE TABLE Trip (

TripID INT AUTO\_INCREMENT PRIMARY KEY,

BookingID INT,

AmbulanceID INT,

PickupTime DATETIME,

DropoffTime DATETIME,

DistanceTraveled VARCHAR(50),

Fare VARCHAR(50),

PaymentStatus VARCHAR(20),

FOREIGN KEY (BookingID) REFERENCES BookingRequest(BookingID),

FOREIGN KEY (AmbulanceID) REFERENCES Ambulance(AmbulanceID)

);

**5.PAYMENTE TABLE**

CREATE TABLE Payment (

PaymentID INT AUTO\_INCREMENT PRIMARY KEY,

TripID INT,

UserID INT,

PaymentMethod VARCHAR(20),

Amount VARCHAR(50),

PaymentDateTime DATETIME,

FOREIGN KEY (TripID) REFERENCES Trip(TripID),

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**6. EMERGENCY CONTANCT TABLE**

CREATE TABLE EmergencyContacts (

ContactID INT AUTO\_INCREMENT PRIMARY KEY,

UserID INT,

ContactName VARCHAR(255),

Relationship VARCHAR(50),

ContactPhoneNumber VARCHAR(20),

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**7.FEEDBACK AND RATINGS TABLE**

CREATE TABLE FeedbackAndRatings (

FeedbackID INT AUTO\_INCREMENT PRIMARY KEY,

TripID INT,

UserID INT,

**Rating\_Stars** VARCHAR(50),

Comments TEXT,

FeedbackDateTime DATETIME,

FOREIGN KEY (TripID) REFERENCES Trip(TripID),

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**8.ADMIN TABLE**

CREATE TABLE Admin (

AdminID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(255),

PhoneNumber VARCHAR(20),

Email VARCHAR(255),

Role VARCHAR(50),

Password VARCHAR(50)

);

**9.NOTIFICATION TABLE**

CREATE TABLE Notification (

NotificationID INT AUTO\_INCREMENT PRIMARY KEY,

UserID INT,

Content TEXT,

Timestamp DATETIME,

Status VARCHAR(20),

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

**10. REPORTS AND ANALYTICS TABLE**

CREATE TABLE ReportsAndAnalytics (

ReportID INT AUTO\_INCREMENT PRIMARY KEY,

ReportType VARCHAR(50),

DateRange VARCHAR(50),

GeneratedBy INT,

ReportContent TEXT,

FOREIGN KEY (GeneratedBy) REFERENCES Admin(AdminID)

);

Q3. write queries to insert data into your tables.

***Data for User table***

INSERT INTO User (Name, PhoneNumber, Email, Address, DateOfBirth, Gender, InsuranceInfo, Password)

VALUES ('John MUGISHA', '+250 788 657 842', 'johnmugisha@gmail.com', 'NR1 Street', '1998-02-11', 'Male', 'UAP', '123456');

***Data for Ambulance table***

INSERT INTO Ambulance (VehicleNumber, Type, Capacity, Equipment, CurrentLatitude, CurrentLongitude, AvailabilityStatus, DriverName, DriverContact, DriverCertification)

VALUES

('RAD-001C', 'Basic', 2, 'First Aid Kit, Oxygen', -1.94918, 30.05836, 'Available', 'Sean MUGABE', '+250731234567', 'Certified EMT'),

('RAG-002D', 'Advanced', 4, 'Defibrillator, Ventilator', -1.94956, 30.05974, 'Available', 'Aline KEZA', '+250789876543', 'Paramedic Certified');

***Data for BookingRequest table***

INSERT INTO BookingRequest (UserID, PickupLatitude, PickupLongitude, DestinationLatitude, DestinationLongitude, HospitalName, RequestedTime, PriorityLevel, Status, Reason, AdditionalNotes)

VALUES

(1, -1.949285, 30.059917, -1.932574, 30.061589, 'CHUK', '2023-09-12 08:00:00', 'High', 'Pending', 'Emergency', 'Patient is in critical condition'),

(2, -1.942345, 30.058123, -1.926789, 30.055432, 'Laurier Int Hospital', '2023-09-12 10:30:00', 'Medium', 'Pending', 'Accident', 'Multiple injuries reported');

***Data for Trip table***

INSERT INTO Trip (BookingID, AmbulanceID, PickupTime, DropoffTime, DistanceTraveled, EstimatedArrivalTime, Fare, PaymentStatus)

VALUES

(1, 1, '2023-09-12 10:35:00', '2023-09-12 10:55:00', 5.3, '2023-09-12 10:55:00', 75.00, 'Paid'),

(2, 2, '2023-09-12 11:50:00', '2023-09-12 12:10:00', 4.8, '2023-09-12 12:10:00', 65.00, 'Pending');

***Data for Payment table***

INSERT INTO Payment (TripID, UserID, PaymentMethod, Amount, PaymentDateTime)

VALUES

(1, 1, 'Credit Card', 75.00, '2023-09-12 11:00:00'),

(2, 2, 'Bitcoin', 65.00, '2023-09-12 12:15:00');

***Data for EmergencyContacts table***

INSERT INTO EmergencyContacts (UserID, ContactName, Relationship, ContactPhoneNumber)

VALUES

(1, 'Aline MUGISHA', 'Wife', '+250732345678'),

(2, 'Ane KEZA', 'Sibling', '+250780123456');

***Data for FeedbackAndRatings table***

INSERT INTO FeedbackAndRatings (TripID, UserID, Rating, Comments, FeedbackDateTime)

VALUES

(1, 1, 5, 'Excellent service! Mukomereze aho.', '2023-09-12 11:30:00'),

(2, 2, 4, 'Badutabaye mugihe cyihuse, murakoze.', '2023-09-12 12:30:00');

***Data for Admin table***

INSERT INTO Admin (Name, PhoneNumber, Email, Role)

VALUES

('Jean MURONGO', '+250781111111', 'admin1@gmail.com', 'Administrator'),

('Alice MBONGO', '+250722222222', 'admin2@gmail.com', 'Dispatcher'),

('Laurier Greens RUGWIZA', '+250733333333', 'admin3@gmail.com', 'Coordinator'),

('Martin MIHIGO', '+250782111111', 'admin1@gmail.com', 'Administrator');

Q 4. Write queries to display all the information in your tables.

SELECT \* FROM User;

SELECT \* FROM Ambulance;

SELECT \* FROM BookingRequest;

SELECT \* FROM TRIP;

SELECT \* FROM Payment;

SELECT \* FROM EmergencyContacts;

SELECT \* FROM FeedbackAndRatings;

SELECT \* FROM Notification;

SELECT \* FROM ReportsAndAnalytics;

Q 5. Write a query to update information in any of the two tables of your system

***1.Updating User table and we update a phone number of someone***

UPDATE User

SET PhoneNumber = '+2507234567890'

WHERE UserID = 1;

***2.Updating Admin table and we update a email address of ALICE MBOGO***

UPDATE Admin

SET Email = 'alicembogo@dispatcher.abs.com'

WHERE AdminID = 2;

**SECTION 3**

1. Create a view to insert data into your tables

**create view insert\_Userview AS select\*from User where UserID=4;**

**create view insert\_Adminrview AS select\*from Admin where AdminID=1;**

**create view insert\_Ambulanceview AS select\*from Ambulance where AmbulanceID=3;**

**create view insert\_Tripview AS select\*from Trip where TripID=3;**

**create view insert\_Paymentview AS select\*from Payment where PaymentID=3;**

**create view insert\_BookingRequestview AS select\*from BookingRequest where BookingID=3;**

**create view insert\_EmergencyContactsview AS select\*from EmergencyContacts where ContactID=3;**

1. Create a view to display all the information in your tables.

CREATE VIEW AllInformationView AS

SELECT

u.UserID,

u.Name AS UserName,

u.PhoneNumber,

u.Email,

u.Address,

u.DateOfBirth,

u.Gender,

u.MedicalHistory,

u.InsuranceInfo,

a.AmbulanceID,

a.VehicleNumber,

a.Type AS AmbulanceType,

a.Capacity AS AmbulanceCapacity,

a.Equipment AS AmbulanceEquipment,

a.CurrentLatitude AS AmbulanceLatitude,

a.CurrentLongitude AS AmbulanceLongitude,

a.AvailabilityStatus AS AmbulanceStatus,

a.DriverName AS AmbulanceDriverName,

a.DriverContact AS AmbulanceDriverContact,

a.DriverCertification AS AmbulanceDriverCertification,

br.BookingID,

br.PickupLatitude AS PickupLatitude,

br.PickupLongitude AS PickupLongitude,

br.DestinationLatitude AS DestinationLatitude,

br.DestinationLongitude AS DestinationLongitude,

br.HospitalName AS DestinationHospitalName,

br.RequestedTime AS BookingRequestedTime,

br.PriorityLevel AS BookingPriority,

br.Status AS BookingStatus,

br.Reason AS BookingReason,

br.AdditionalNotes AS BookingNotes,

t.TripID,

t.PickupTime AS TripPickupTime,

t.DropoffTime AS TripDropoffTime,

t.DistanceTraveled AS TripDistanceTraveled,

t.EstimatedArrivalTime AS TripEstimatedArrivalTime,

t.Fare AS TripFare,

t.PaymentStatus AS TripPaymentStatus,

p.PaymentID,

p.PaymentMethod,

p.Amount AS PaymentAmount,

p.PaymentDateTime AS PaymentDateTime,

ec.ContactID AS EmergencyContactID,

ec.ContactName AS EmergencyContactName,

ec.Relationship AS EmergencyContactRelationship,

ec.ContactPhoneNumber AS EmergencyContactPhoneNumber,

far.FeedbackID AS FeedbackID,

far.Rating AS FeedbackRating,

far.Comments AS FeedbackComments,

far.FeedbackDateTime AS FeedbackDateTime,

n.NotificationID AS NotificationID,

n.Content AS NotificationContent,

n.Timestamp AS NotificationTimestamp,

n.Status AS NotificationStatus,

rra.ReportID AS ReportID,

rra.ReportType AS ReportType,

rra.DateRange AS ReportDateRange,

rra.GeneratedBy AS ReportGeneratedBy,

rra.ReportContent AS ReportContent

FROM

User u

LEFT JOIN

Ambulance a ON u.UserID = a.AmbulanceID

LEFT JOIN

BookingRequest br ON u.UserID = br.UserID

LEFT JOIN

Trip t ON br.BookingID = t.BookingID

LEFT JOIN

Payment p ON t.TripID = p.TripID

LEFT JOIN

EmergencyContacts ec ON u.UserID = ec.UserID

LEFT JOIN

FeedbackAndRatings far ON t.TripID = far.TripID

LEFT JOIN

Notification n ON u.UserID = n.UserID

LEFT JOIN

ReportsAndAnalytics rra ON u.UserID = rra.GeneratedBy;

1. Create a view to update information in any of the two tables of your system.

UPDATE User

SET PhoneNumber = '+250787777777'

WHERE UserID = 4;

UPDATE Admin

SET Email = 'martinmihigo@admin.abs.com'

WHERE AdminID = 4;

1. Create a view to delete data in any two of your tables according to any simple condition of your choice.

DELETE FROM User

WHERE UserID = 4;

DELETE FROM Admin

WHERE AdminID = 4;

1. In your database, create one view of your choice that considers sub-query.

CREATE VIEW HighPriorityBookings AS

SELECT \*

FROM BookingRequest

WHERE PriorityLevel = 'High';

SELECT \*

FROM HighPriorityBookings;

**SECTIO 4**

1. Create a stored procedure to insert data into your tables.

DELIMITER //

CREATE PROCEDURE InsertUserData(

IN p\_Name VARCHAR(255),

IN p\_PhoneNumber VARCHAR(20),

IN p\_Email VARCHAR(255),

IN p\_Address VARCHAR(255),

IN p\_DateOfBirth DATE,

IN p\_Gender VARCHAR(10),

IN p\_MedicalHistory TEXT,

IN p\_InsuranceInfo TEXT

)

BEGIN

INSERT INTO User (Name, PhoneNumber, Email, Address, DateOfBirth, Gender, MedicalHistory, InsuranceInfo)

VALUES (p\_Name, p\_PhoneNumber, p\_Email, p\_Address, p\_DateOfBirth, p\_Gender, p\_MedicalHistory, p\_InsuranceInfo);

END;

//

DELIMITER ;

***Call the stored procedure to insert user data***

CALL InsertUserData('John RUREMA', '+250732345678', 'john@gmail.com', 'Kigali, Rwanda', '1990-01-15', 'Male', 'Heart Attack', 'Greens Insurance');

1. Create a stored procedure to display all the information in your tables.

DELIMITER //

CREATE PROCEDURE DisplayAllInformation()

BEGIN

SELECT \* FROM User;

SELECT \* FROM Ambulance;

SELECT \* FROM BookingRequest;

SELECT \* FROM Trip;

SELECT \* FROM Payment;

SELECT \* FROM EmergencyContacts;

SELECT \* FROM FeedbackAndRatings;

SELECT \* FROM Admin;

SELECT \* FROM Notification;

SELECT \* FROM ReportsAndAnalytics;

END;

//

DELIMITER ;

1. Create a stored procedure to update information in any of the two tables of your system.

DELIMITER //

CREATE PROCEDURE UpdateUserAndEmergencyContacts(

IN p\_UserID INT,

IN p\_NewPhoneNumber VARCHAR(20),

IN p\_NewEmail VARCHAR(255),

IN p\_ContactID INT,

IN p\_NewContactName VARCHAR(255),

IN p\_NewContactPhoneNumber VARCHAR(20)

)

BEGIN

UPDATE User

SET

PhoneNumber = p\_NewPhoneNumber,

Email = p\_NewEmail

WHERE

UserID = p\_UserID;

UPDATE EmergencyContacts

SET

ContactName = p\_NewContactName,

ContactPhoneNumber = p\_NewContactPhoneNumber

WHERE

ContactID = p\_ContactID;

END;

//

DELIMITER ;

Example of An Update from this procedure is as follow:

CALL UpdateUserAndEmergencyContacts(1, '+250780116574', 'joh@gmail.com', 2, 'Joh KAREMERA', '+250784576456');

1. Create a stored procedure to delete data in any two of your tables according to any simple condition of your choice.

DELIMITER //

CREATE PROCEDURE DeleteDataFromTables()

BEGIN

DELETE FROM BookingRequest

WHERE PriorityLevel = 'Low';

DELETE FROM FeedbackAndRatings

WHERE Rating < 3;

END;

//

DELIMITER ;

1. In your database, stored the procedure view of your choice that considers subquery

***In this stored procedure named "GetHighPriorityBookings," we use a subquery to retrieve BookingRequest records with a priority level of 'High.***

DELIMITER //

CREATE PROCEDURE GetHighPriorityBookings()

BEGIN

SELECT \*

FROM BookingRequest

WHERE PriorityLevel = 'High';

SELECT User.\*

FROM User

WHERE UserID IN (SELECT UserID FROM BookingRequest WHERE PriorityLevel = 'High');

END;

//

DELIMITER ;

**SECTION 5**

1. Create after inserting triggers for any two tables of your choice.

**we will create triggers for the "User" and "FeedbackAndRatings" tables:**

***After-Insert Trigger for the "User" Table:***

DELIMITER //

CREATE TRIGGER UserAfterInsert

AFTER INSERT ON User

FOR EACH ROW

BEGIN

INSERT INTO Notification (UserID, Content, Timestamp, Status)

VALUES (NEW.UserID, 'A new user has been added.', NOW(), 'New User');

END;

//

DELIMITER ;

***After-Insert Trigger for the "FeedbackAndRatings" Table:***

DELIMITER //

CREATE TRIGGER FeedbackAfterInsert

AFTER INSERT ON FeedbackAndRatings

FOR EACH ROW

BEGIN

UPDATE Trip

SET Rating = (SELECT AVG(Rating) FROM FeedbackAndRatings WHERE TripID = NEW.TripID)

WHERE TripID = NEW.TripID;

END;

//

DELIMITER ;

To view if it worked properly, we are going to add some data into those tables and see if its going to be in notification.

ADD NEW USER TO SEE IF WE GET NOTIFICATION

INSERT INTO User (Name, PhoneNumber, Email, Address, DateOfBirth, Gender, MedicalHistory, InsuranceInfo)

VALUES

('Alain MIKWEGE', '+250723456700', 'alainmikwege@gmail.com', 'Kigali, Rwanda', '1990-11-25', 'Male', 'Headache', 'Greens Insurance');

ADD NEW BOOKING REQUEST

INSERT INTO BookingRequest (UserID, PickupLatitude, PickupLongitude, DestinationLatitude, DestinationLongitude, HospitalName, RequestedTime, PriorityLevel, Status, Reason, AdditionalNotes)

VALUES (3, -1.942345, 30.058123, -1.926789, 30.055432, 'Laurier Int Hospital', '2023-09-12 10:35:10', 'High', 'Pending', 'Accident', 'Patient had a panic');

ADD NEW TRIP

INSERT INTO Trip (BookingID, AmbulanceID, PickupTime, DropoffTime, DistanceTraveled, EstimatedArrivalTime, Fare, PaymentStatus)

VALUES

(3, 2, '2023-09-12 10:52:10', '2023-09-12 11:37:19', 5.8, '2023-09-12 12:10:00', 70.50, 'Paid');

ADD NEW PAYMENT

INSERT INTO Payment (TripID, UserID, PaymentMethod, Amount, PaymentDateTime)

VALUES

(3, 3, 'Bitcoin', 70.50, '2023-09-12 12:15:00');

1. Create after-update triggers for any two tables of your choice.

After-update triggers for two tables: "User" and "Ambulance." These triggers will perform actions after an update occurs in these tables.

***After-Update Trigger for the "User" Table:***

DELIMITER //

CREATE TRIGGER UserAfterUpdate

AFTER UPDATE ON User

FOR EACH ROW

BEGIN

INSERT INTO Notification (UserID, Content, Timestamp, Status)

VALUES (NEW.UserID, CONCAT('User information updated: ', NEW.Name), NOW(), 'User Updated');

END;

//

DELIMITER ;

***After-Update Trigger for the "Ambulance" Table:***

DELIMITER //

CREATE TRIGGER AmbulanceAfterUpdate

AFTER UPDATE ON Ambulance

FOR EACH ROW

BEGIN

IF NEW.AvailabilityStatus = 'Unavailable' THEN

INSERT INTO Notification (UserID, Content, Timestamp, Status)

VALUES (1, 'Ambulance is now unavailable: ' || NEW.VehicleNumber, NOW(), 'Ambulance Unavailable');

END IF;

END;

//

DELIMITER ;

1. Create after deleting triggers for any two tables of your choice.

***After-Delete Trigger for the "User" Table:***

DELIMITER //

CREATE TRIGGER UserAfterDelete

AFTER DELETE ON User

FOR EACH ROW

BEGIN

INSERT INTO Notification (UserID, Content, Timestamp, Status)

VALUES (OLD.UserID, CONCAT('User deleted: ', OLD.Name), NOW(), 'User Deleted');

END;

//

***This trigger, named "UserAfterDelete," will be executed after a record is deleted from the "User" table. It logs the deletion in the "Notification" table.***

***After-Delete Trigger for the "BookingRequest" Table:***

DELIMITER //

CREATE TRIGGER BookingRequestAfterDelete

AFTER DELETE ON BookingRequest

FOR EACH ROW

BEGIN

UPDATE Trip

SET Status = 'Canceled'

WHERE BookingID = OLD.BookingID;

END;

//

**SECTION 6**

1. Create a user with your name as username and your student number as password and grant all privileges to the created user.

CREATE USER 'laurier'@'localhost' IDENTIFIED BY '222003068';

GRANT ALL PRIVILEGES ON \*.\* TO 'laurier'@'localhost';

FLUSH PRIVILEGES;

1. Create a user with your "names\_semi" as username and your student number as password and give him insert, update, and delete privileges to the created user.

CREATE USER 'laurier\_semi'@'localhost' IDENTIFIED BY '222003068';

GRANT INSERT, UPDATE, DELETE ON \*.\* TO 'laurier\_semi'@'localhost';

FLUSH PRIVILEGES;

//second user

CREATE USER '222003068'@'localhost' IDENTIFIED BY '222003068';

GRANT INSERT, UPDATE, DELETE ON \*.\* TO '222003068'@'localhost';

FLUSH PRIVILEGES;

1. Revoke insert privileges to the last user you created.

REVOKE INSERT ON \*.\* FROM 'laurier\_semi'@'localhost';

FLUSH PRIVILEGES;