Ryan, Alaina, Meerab, Omar

Database LIA Project – Deliverable 3

CREATE TABLE Clients (

ClientID INT PRIMARY KEY,

FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

ContactInformation VARCHAR(255)

);

CREATE TABLE Dentists (

DentistID INT PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Specialization VARCHAR(255) NOT NULL,

ContactInformation VARCHAR(255)

);

CREATE TABLE Appointments (

AppointmentID INT PRIMARY KEY,

AppointmentType VARCHAR(255),

DateTime DATETIME,

ClientID INT,

DentistID INT,

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),

FOREIGN KEY (DentistID) REFERENCES Dentists(DentistID)

);

CREATE TABLE Visits (

VisitID INT PRIMARY KEY,

DateTime DATETIME,

Diagnosis VARCHAR(255),

Treatment VARCHAR(255),

AppointmentID INT,

ClientID INT,

DentistID INT,

BillAmount DECIMAL(10, 2),

BillBalance DECIMAL(10, 2),

FOREIGN KEY (AppointmentID) REFERENCES Appointments(AppointmentID),

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),

FOREIGN KEY (DentistID) REFERENCES Dentists(DentistID)

);

CREATE TABLE ClientPayment (

CPaymentID INT PRIMARY KEY,

InsuranceID INT,

InstallmentAmount DECIMAL(10, 2),

InstallmentDeductible DECIMAL(10, 2),

IsBasisForClaim VARCHAR(3) CHECK (IsBasisForClaim IN ('Yes', 'No')),

VisitID INT,

FOREIGN KEY (VisitID) REFERENCES Visits(VisitID)

);

CREATE TABLE InsuranceCompanyPayment (

InsuranceID INT,

PaymentID INT,

ClientID INT,

IsPaying VARCHAR(3) CHECK (IsPaying IN ('Yes','No')),

PRIMARY KEY (InsuranceID),

CPaymentID INT,

FOREIGN KEY (CPaymentID) REFERENCES ClientPayment(CPaymentID),

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID)

);

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY,

Amount DECIMAL(10, 2),

DateTime DATETIME,

CPaymentID INT,

PaymentMethod VARCHAR(255),

NumOfInstallments INT DEFAULT 1 CHECK (NumOfInstallments > 0),

ClientID INT,

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),

FOREIGN KEY (CPaymentID) REFERENCES ClientPayment(CPaymentID)

);

CREATE TABLE InsuranceCompany (

InsuranceCompanyID INT PRIMARY KEY,

Name VARCHAR(255),

ContactInformation VARCHAR(255),

VisitID INT,

FOREIGN KEY (VisitID) REFERENCES Visits(VisitID)

);

INSERT INTO Clients (ClientID, FirstName, LastName, ContactInformation) VALUES

(101, 'Alice', 'Johnson', 'alice@gmail.com'),

(102, 'Bob', 'Smith', 'bob@gmail.com'),

(103, 'Emily', 'Brown', 'emily@gmail.com'),

(104, 'John', 'Doe', 'john@yahoo.com'),

(105, 'Sarah', 'Lee', 'sarah@hotmail.com');

INSERT INTO Dentists (DentistID, Name, Specialization, ContactInformation) VALUES

(201, 'Dr. Davis', 'Orthodontics', 'dr.davis@hotmail.com'),

(202, 'Dr. Martinez', 'Pediatric Dentistry', 'dr.martinez@yahoo.com'),

(203, 'Dr. Lee', 'Endodontics', 'dr.lee@gmail.com'),

(204, 'Dr. Wilson', 'Periodontics', 'dr.wilson@hotmail.com'),

(205, 'Dr. Rodriguez', 'Oral Surgery', 'dr.rodriguez@yahoo.com');

INSERT INTO Appointments (AppointmentID, AppointmentType, DateTime, ClientID, DentistID) VALUES

(301, 'Regular Checkup', '2024-03-23 10:00:00', 101, 201),

(302, 'Emergency', '2024-03-24 14:15:00', 102, 202),

(303, 'Regular Checkup', '2024-04-25 15:00:00', 104, 203),

(304, 'Braces Adjustment', '2024-10-28 11:00:00', 105, 204),

(305, 'Consultation', '2024-11-27 15:00:00', 103, 205);

INSERT INTO Appointments (AppointmentID, AppointmentType, DateTime, ClientID, DentistID) VALUES

(306, 'Braces Adjustment', '2024-10-28 11:00:00', 105, 204);--to add for two apointments on the same day

INSERT INTO Visits (VisitID, DateTime, Diagnosis, Treatment, AppointmentID, ClientID, DentistID, BillAmount, BillBalance) VALUES

(401, '2024-03-23 10:00:00', 'Teeth cleaning', 'Checkup', 301, 101, 201, 200.00, 0.00),

(402, '2024-03-24 14:15:00', 'Toothache', 'Extraction', 302, 102, 202, 350.00, 150.00),

(403, '2024-04-25 15:00:00', 'Braces adjustment', 'Tightening', 303, 104, 203, 150.00, 50.00),

(404, '2024-03-26 11:00:00', 'Checkup', 'Checkup', 304, 105, 204, 100.00, 0.00),

(405, '2024-03-27 15:00:00', 'Wisdom tooth removal', 'Surgery', 305, 103, 205, 120.00, 0.00);

INSERT INTO ClientPayment (CPaymentID, InsuranceID, InstallmentAmount, InstallmentDeductible, IsBasisForClaim, VisitID) VALUES

(501, 801, 50.00, 25.00, 'Yes', 401),

(502, 802, 75.00, 30.00, 'No', 402),

(503, 803 , 100.00, 40.00, 'Yes', 403),

(504, 804, 55.00, 20.00, 'Yes', 404),

(505, 805 , 80.00, 35.00, 'No', 405);

INSERT INTO Payments (PaymentID, Amount, DateTime, CPaymentID, PaymentMethod, NumOfInstallments) VALUES

(701, 150.00, '2024-03-01', 501, 'Credit Card', 3),

(702, 200.00, '2024-03-05', 502, 'Bank Transfer', 2),

(703, 125.00, '2024-03-10', 503, 'Cash', 1),

(704, 175.00, '2024-03-15', 504, 'PayPal', 4),

(705, 100.00, '2024-03-20', 505, 'Cheque', 2);

INSERT INTO InsuranceCompanyPayment (InsuranceID, PaymentID, ClientID, IsPaying) VALUES

(801, 701, 101, 'Yes'),

(802, 702, 102, 'No'),

(803, 703, 103, 'Yes'),

(804, 704, 104, 'No'),

(805, 705, 105, 'Yes');

INSERT INTO InsuranceCompany (InsuranceCompanyID, Name, ContactInformation, VisitID) VALUES

(601, 'SafetyFirst Insurance', 'info@safetyfirstinsurance.com', 401),

(602, 'Liberty Mutual', 'contact@libertymutual.com', 402),

(603, 'Acme Insurance Co.', 'support@acmeinsuranceco.com', 403),

(604, 'Guardian Insurance', 'help@guardianinsurance.com', 404),

(605, 'Pioneer Insurance', 'info@pioneerinsurance.com', 405);

1. Create a procedure that selects all clients who have booked an appointment with a dentist with a specific specialization (using EXISTS)

CREATE PROCEDURE clientAppointmentWithDentistSpecialization

@Specialization VARCHAR(50)

AS

BEGIN

SELECT c.\*, a.DentistID

FROM Clients c

INNER JOIN Appointments a ON a.ClientID = c.ClientID

WHERE EXISTS (

SELECT \*

FROM Dentists d

WHERE a.DentistID = d.DentistID AND d.Specialization = @Specialization

);

END;

EXEC clientAppointmentWithDentistSpecialization orthodontics;

1. Create a function that gets and shows all clients whose emails end with ‘@gmail.com’.

CREATE FUNCTION gmailClientsTable()

RETURNS @gmailClients TABLE (ClientID INT PRIMARY KEY, FirstName VARCHAR(50),LastName VARCHAR(50), ContactInformation VARCHAR(50))

AS

BEGIN

INSERT INTO @gmailClients(ClientID, FirstName, LastName, ContactInformation)

SELECT \*

FROM Clients

WHERE ContactInformation LIKE '%@gmail.com';

RETURN;

END;

SELECT \* FROM gmailClientsTable();

1. Create a sequence beginning with a value of 901 and that increments by 1. Then begin a transaction that updates all the PaymentIDs of the Payment table with the sequence. If any of them equal 904, then the transaction will be rollbacked; else, transaction commits.

CREATE SEQUENCE seq

START WITH 901

INCREMENT BY 1

NO cycle;

BEGIN TRANSACTION t

UPDATE Payments

SET PaymentID = NEXT VALUE FOR seq

IF EXISTS(SELECT \* FROM Payments WHERE PaymentID = 904)

BEGIN

PRINT('PaymentID 904 detected, rollbacking transaction');

ROLLBACK TRANSACTION t;

END;

ELSE

BEGIN

PRINT('Transaction successful');

COMMIT TRANSACTION t;

END;

SELECT \* FROM Payments;

1. Create an index for performance on the ClientID foreign key in the Appointments table

CREATE INDEX idx\_ClientID ON Appointments (ClientID);

1. Create a view that shows client details with their appointment information.

CREATE VIEW ClientAppointments AS

SELECT

C.ClientID,

C.FirstName,

C.LastName,

C.ContactInformation AS ClientContact,

A.AppointmentID,

A.AppointmentType,

A.DateTime AS AppointmentDateTime,

A.DentistID,

D.Name AS DentistName,

D.Specialization AS DentistSpecialization,

D.ContactInformation AS DentistContact

FROM

Clients C

JOIN

Appointments A ON C.ClientID = A.ClientID

JOIN

Dentists D ON A.DentistID = D.DentistID;

SELECT \* FROM ClientAppointments;

1. Create a trigger that prevents creation, deletion, and alteration of table.

CREATE TRIGGER triggerRestrictDDLEvents

ON DATABASE

FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

BEGIN

PRINT 'Action create, alter or drop a table not allowed'

ROLLBACK TRANSACTION

END;

1. Retrieve the number of appointments made per month

SELECT MONTH(DateTime) AS Month, COUNT(\*) AS NumAppointments

FROM Appointments

GROUP BY MONTH(DateTime);

1. Create a procedure that calculates the total bill amount for given client:

CREATE PROCEDURE CalculateTotalBillAmountForClient

@client\_id INT

AS

BEGIN

DECLARE @total\_bill DECIMAL(10, 2);

-- Calculate total bill amount for the client

SELECT @total\_bill = SUM(BillAmount)

FROM Visits

WHERE ClientID = @client\_id;

-- Display the total bill amount

SELECT @total\_bill AS TotalBillAmount;

END;

EXEC CalculateTotalBillAmountForClient @client\_id = 101;