

ASSIGNMENT (IV)

**Art Gallery Management System**

**Question 1 :**

**Stored Procedures**

* **Sales Report by Date Range**

-- Procedure For Sales Report By Range

CREATE PROCEDURE sp\_SalesReportByDateRange

@StartDate DATE,

@EndDate DATE

AS

BEGIN

SELECT

S.SaleID,

S.SaleDate,

A.Title AS ArtworkTitle,

U.FirstName + ' ' + U.LastName AS BuyerName,

S.Price,

S.ArtistCommission

FROM

Sales S

JOIN Artwork A ON S.ArtID = A.ArtID

JOIN Users U ON S.UserID = U.UserID

WHERE

S.SaleDate BETWEEN @StartDate AND @EndDate

ORDER BY

S.SaleDate;

END;

GO



* **Report on Loans Taken by User**

-- Procedure For Loan Report

CREATE PROCEDURE sp\_LoanReportByUser

@UserID INT

AS

BEGIN

SELECT

L.LoanID,

L.LoanAmount,

L.DurationInDays,

L.ExpiryDate,

L.TransportTracking,

L.MonitoringStatus,

A.Title AS ArtworkTitle

FROM

Loan L

JOIN Artwork A ON L.ArtID = A.ArtID

WHERE

L.UserID = @UserID

ORDER BY

L.ExpiryDate DESC;

END;

GO

**A screenshot of a box

Description automatically generated**

* **Report on Total Sales and Commission by Artist**

-- Procedure For Artist Contribution Report

CREATE PROCEDURE sp\_TotalSalesAndCommissionByArtist

@StartDate DATE,

@EndDate DATE

AS

BEGIN

SELECT

AR.UserID,

U.FirstName + ' ' + U.LastName AS ArtistName,

SUM(S.Price) AS TotalSales,

SUM(S.ArtistCommission) AS TotalCommission

FROM

Sales S

JOIN Artwork A ON S.ArtID = A.ArtID

JOIN Artist AR ON A.AID = AR.AID

JOIN Users U ON AR.UserID = U.UserID

WHERE

S.SaleDate BETWEEN @StartDate AND @EndDate

GROUP BY

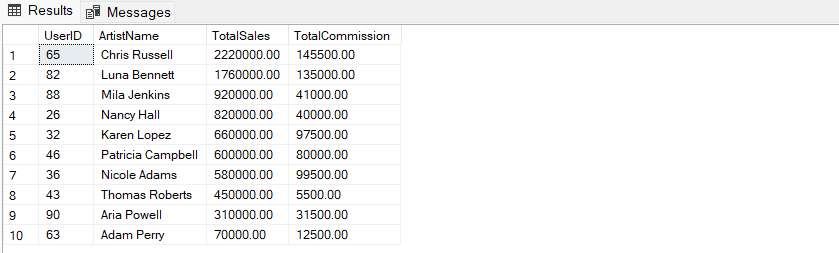
AR.UserID, U.FirstName, U.LastName

ORDER BY

TotalSales DESC;

END;

GO

****

* **Report on Exhibition for a Gallery**

-- Procedure For Exhibition Report For A Gallery

CREATE PROCEDURE sp\_ExhibitionReportByGallery

@GalleryID INT

AS

BEGIN

SELECT

E.EID,

E.EName,

E.StartDate,

E.EndDate,

E.Location,

E.Theme,

A.Title AS ArtworkTitle

FROM

Exhibition E

JOIN Artwork A ON E.ArtID = A.ArtID

WHERE

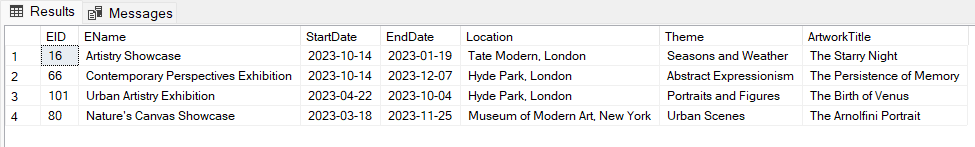
E.GID = @GalleryID

ORDER BY

E.StartDate DESC;

END;

GO

****

**Complex Views**

* **Artist Sales Report**

-- Artist Sales Report Through Complex View

CREATE VIEW SalesReportByArtist AS

SELECT

AR.UserID AS ArtistID,

CONCAT(U.FirstName, ' ', U.LastName) AS ArtistName,

A.Title AS ArtworkTitle,

S.SaleDate,

S.Price,

S.ArtistCommission

FROM

Sales S

JOIN

Artwork A ON S.ArtID = A.ArtID

JOIN

Artist AR ON A.AID = AR.AID

JOIN

Users U ON AR.UserID = U.UserID;

GO



* **Exhibition Details Report**

-- Exhibition Details Report

CREATE VIEW ExhibitionDetails AS

SELECT

E.EName AS ExhibitionName,

E.StartDate,

E.EndDate,

G.GalleryName,

G.Location,

STRING\_AGG(A.Title, ', ') AS ArtworkTitles

FROM

Exhibition E

JOIN

Gallery G ON E.GID = G.GID

JOIN

Artwork A ON E.ArtID = A.ArtID

GROUP BY

E.EName, E.StartDate, E.EndDate, G.GalleryName, G.Location;

GO

****

* **Loan Status Report**

--Loan Status Report

CREATE VIEW LoanStatusReport AS

SELECT

L.LoanID,

A.Title AS ArtworkTitle,

L.LoanAmount,

L.DurationInDays,

L.ExpiryDate,

CONCAT(U.FirstName, ' ', U.LastName) AS BorrowerName,

L.MonitoringStatus

FROM

Loan L

JOIN

Artwork A ON L.ArtID = A.ArtID

JOIN

Users U ON L.UserID = U.UserID

WHERE

L.ExpiryDate >= GETDATE(); -- Filter for active loans

GO

****

* **Artist Performance Report**

CREATE VIEW ArtistPerformanceReport AS

SELECT

A.AID AS ArtistID,

CONCAT(U.FirstName, ' ', U.LastName) AS ArtistName,

COUNT(W.ArtID) AS UploadedArtworks,

SUM(S.Price) AS TotalRevenue,

SUM(S.ArtistCommission) AS TotalCommissionEarned

FROM

Artist A

JOIN

Users U ON A.UserID = U.UserID

LEFT JOIN

Artwork W ON A.AID = W.AID

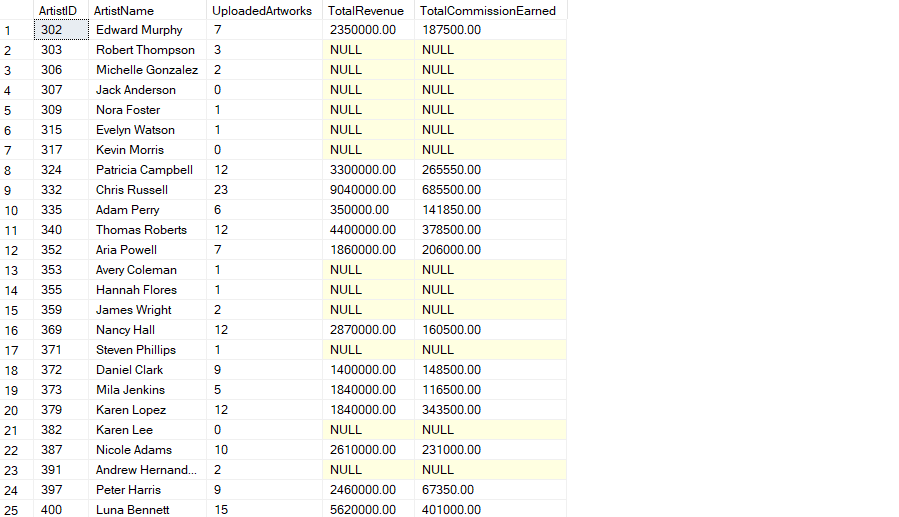
LEFT JOIN

Sales S ON W.ArtID = S.ArtID

GROUP BY

A.AID, U.FirstName, U.LastName;

GO

****

* **Financial Summary Report**

CREATE VIEW FinancialSummaryReport AS

SELECT

SUM(S.Price) AS TotalSalesRevenue,

SUM(S.ArtistCommission) AS TotalArtistCommissions,

SUM(L.LoanAmount) AS TotalLoanIncome

FROM

Sales S

JOIN

Users U ON S.UserID = U.UserID

LEFT JOIN

Loan L ON S.ArtID = L.ArtID;

GO

****

**Materialized Views**

* **Report on Total Sales By Artist**

-- MATERIALIZE VIEW Total sales by Artist

CREATE VIEW Materialized\_ArtistSalesSummary

WITH SCHEMABINDING

AS

SELECT

AR.UserID AS ArtistID,

CONCAT(U.FirstName, ' ', U.LastName) AS ArtistName,

AR.SocialMediaProfile,

SUM(S.Price) AS TotalSales

FROM

dbo.Sales S

JOIN

dbo.Artwork A ON S.ArtID = A.ArtID

JOIN

dbo.Artist AR ON A.AID = AR.AID

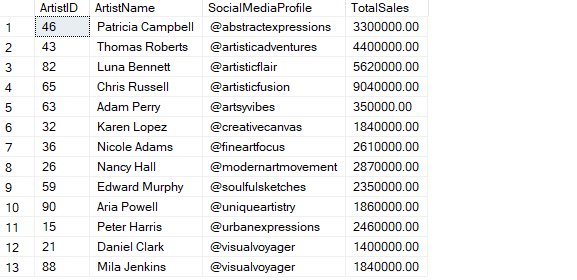
JOIN

dbo.Users U ON AR.UserID = U.UserID

GROUP BY

AR.UserID, U.FirstName, U.LastName, AR.SocialMediaProfile;

GO

****

* **Report For Available Artwork**

-- Report For Available Artwork

CREATE VIEW Materialized\_ArtworkAvailability

WITH SCHEMABINDING

AS

SELECT

A.ArtID,

A.Title AS ArtworkTitle,

CONCAT(U.FirstName, ' ', U.LastName) AS ArtistName,

AR.SocialMediaProfile,

A.Availability

FROM

dbo.Artwork A

JOIN

dbo.Artist AR ON A.AID = AR.AID

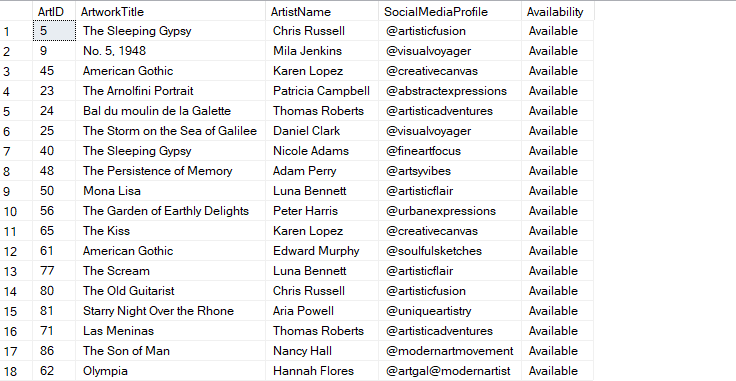
JOIN

dbo.Users U ON AR.UserID = U.UserID

WHERE

A.Availability = 'Available';

GO

****

* **Report For Exhibition Details**

-- Report For Exhibition Details

CREATE VIEW Materialized\_ExhibitionDetails

WITH SCHEMABINDING

AS

SELECT

E.EID,

E.EName AS ExhibitionName,

E.StartDate,

E.EndDate,

G.GalleryName,

G.Location,

A.Title AS ArtworkTitle,

A.Availability

FROM

dbo.Exhibition E

JOIN

dbo.Gallery G ON E.GID = G.GID

JOIN

dbo.Artwork A ON E.ArtID = A.ArtID;

GO

****

**Question 2 :**

**Insertion Procedure**

GO

CREATE PROCEDURE InsertUserData

@UserID INT,

@UserEmail VARCHAR(255),

@Username VARCHAR(100),

@Password VARCHAR(255),

@Role VARCHAR(100),

@FirstName VARCHAR(100),

@LastName VARCHAR(100),

@Address VARCHAR(255),

@PhoneNo VARCHAR(20),

@RowsAffected INT OUTPUT

AS

BEGIN

INSERT INTO Users (UserID, UserEmail, Username, Password, Role, FirstName, LastName, Address, PhoneNo)

VALUES (@UserID, @UserEmail, @Username, @Password, @Role, @FirstName, @LastName, @Address, @PhoneNo);

SET @RowsAffected = @@ROWCOUNT;

END;

GO

DECLARE @RowsAffected INT;

EXEC InsertUserData

@UserID = 101,

@UserEmail = 'user101@example.com',

@Username = 'user101',

@Password = 'user101!',

@Role = 'Staff',

@FirstName = 'Rebecca',

@LastName = 'Johnson',

@Address = '101 Maple Street',

@PhoneNo = '123-456-7101',

@RowsAffected = @RowsAffected OUTPUT;

****

CREATE PROCEDURE InsertArtwork

@AID INT,

@Title VARCHAR(255),

@Medium VARCHAR(100),

@Dimensions VARCHAR(100),

@DateCompleted DATE,

@Availability VARCHAR(20),

@Price DECIMAL(10, 2)

AS

BEGIN

INSERT INTO Artwork (AID, Title, Medium, Dimensions, DateCompleted, Availability, Price)

VALUES (@AID, @Title, @Medium, @Dimensions, @DateCompleted, @Availability, @Price);

END;

GO

EXEC InsertArtwork

@AID = 101,

@Title = 'Sunset Over the Ocean',

@Medium = 'Oil on Canvas',

@Dimensions = '24x36 inches',

@DateCompleted = '2023-01-01',

@Availability = 'Available',

@Price = 1500.00;

CREATE PROCEDURE InsertGallery

@UserID INT,

@GalleryName VARCHAR(255),

@Status VARCHAR(50),

@Location VARCHAR(255)

AS

BEGIN

INSERT INTO Gallery (UserID, GalleryName, Status, Location)

VALUES (@UserID, @GalleryName, @Status, @Location);

END;

GO

EXEC InsertGallery

@UserID = 1,

@GalleryName = 'Modern Art Gallery',

@Status = 'Open',

@Location = '123 Art St';

**Updation Procedure**

CREATE PROCEDURE UpdateArtworkData

@ArtID INT,

@Availability VARCHAR(20),

@Price DECIMAL(10, 2)

AS

BEGIN

SELECT \* FROM Artwork WHERE ArtID = @ArtID AND Availability = 'Available';

UPDATE Artwork

SET Availability = @Availability,

Price = @Price

WHERE ArtID = @ArtID;

SELECT \* FROM Artwork WHERE ArtID = @ArtID;

DECLARE @RowsAffected INT;

SET @RowsAffected = @@ROWCOUNT;

SELECT @RowsAffected AS RowsAffected;

END;

GO

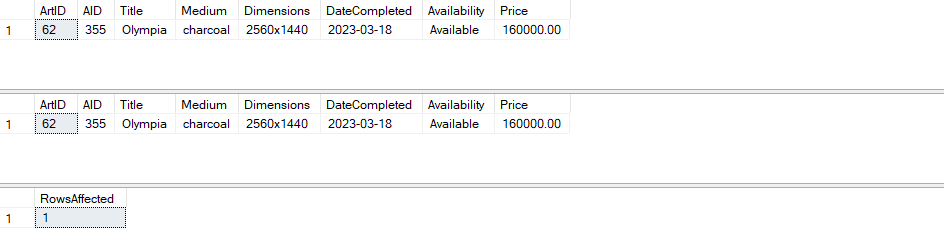
DECLARE @RowsAffected INT;

EXEC UpdateArtworkData

@ArtID = 62,

@Availability = 'Available',

@Price = 160000;

****

CREATE PROCEDURE UpdateLoanDetails

@LoanID INT,

@LoanAmount DECIMAL(10, 2) = NULL,

@DurationInDays INT = NULL,

@ExpiryDate DATE = NULL,

@TransportTracking VARCHAR(100) = NULL,

@MonitoringStatus VARCHAR(50) = NULL

AS

BEGIN

SET NOCOUNT ON;

IF NOT EXISTS (SELECT 1 FROM Loan WHERE LoanID = @LoanID)

BEGIN

RAISERROR('LoanID %d does not exist.', 16, 1, @LoanID);

RETURN;

END

UPDATE Loan

SET

LoanAmount = COALESCE(@LoanAmount, LoanAmount),

DurationInDays = COALESCE(@DurationInDays, DurationInDays),

ExpiryDate = COALESCE(@ExpiryDate, ExpiryDate),

TransportTracking = COALESCE(@TransportTracking, TransportTracking),

MonitoringStatus = COALESCE(@MonitoringStatus, MonitoringStatus)

WHERE

LoanID = @LoanID;

END;

EXEC UpdateLoanDetails

@LoanID = 68,

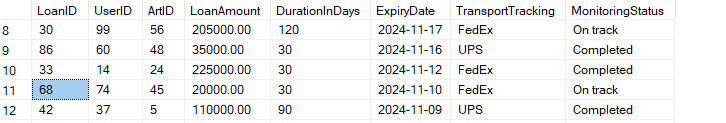
@LoanAmount = 20000.00,

@DurationInDays = 30,

@ExpiryDate = '2024-11-10',

@TransportTracking = 'FedEx',

@MonitoringStatus = 'On track';

****

CREATE PROCEDURE UpdateGalleryDetails

@GID INT,

@GalleryName VARCHAR(255),

@Status VARCHAR(50),

@Location VARCHAR(255)

AS

BEGIN

UPDATE Gallery

SET GalleryName = @GalleryName,

Status = @Status,

Location = @Location

WHERE GID = @GID;

END;

GO

EXEC UpdateGalleryDetails

@GID = 1,

@GalleryName = 'Modern Art Gallery',

@Status = 'Open',

@Location = '123 Art St';

**Deletion Procedure**

GO

CREATE PROCEDURE DeleteUserDataAndRelated

@UserID INT,

@RowsAffected INT OUTPUT

AS

BEGIN

DECLARE @TotalRowsAffected INT = 0;

DELETE FROM Artist

WHERE UserID = @UserID;

SET @TotalRowsAffected = @TotalRowsAffected + @@ROWCOUNT;

DELETE FROM Gallery

WHERE UserID = @UserID;

SET @TotalRowsAffected = @TotalRowsAffected + @@ROWCOUNT;

DELETE FROM Users

WHERE UserID = @UserID;

SET @TotalRowsAffected = @TotalRowsAffected + @@ROWCOUNT;

SET @RowsAffected = @TotalRowsAffected;

END;

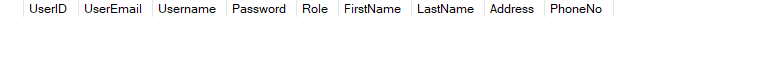
GO

DECLARE @RowsAffected INT;

EXEC DeleteUserDataAndRelated

@UserID = 71,

@RowsAffected = @RowsAffected OUTPUT;

****

CREATE PROCEDURE DeleteArtworkByID

@ArtID INT

AS

BEGIN

DELETE FROM Artwork WHERE ArtID = @ArtID;

END;

GO

EXEC DeleteArtworkByID @ArtID = 2;

CREATE PROCEDURE DeleteArtistByID

@AID INT

AS

BEGIN

DELETE FROM Artist WHERE AID = @AID;

END;

GO

EXEC DeleteArtistByID @AID = 71;

**Triggers**

-- Trigger to prevent deletion of Artwork when its availability is 'Sold' or 'On Loan'

CREATE TRIGGER trg\_PreventArtworkDeletion

ON Artwork

AFTER DELETE

AS

BEGIN

-- Check if any of the deleted artworks are 'Sold' or 'On Loan'

IF EXISTS (SELECT \* FROM deleted WHERE Availability IN ('Sold', 'On Loan'))

BEGIN

-- If any deleted artwork is 'Sold' or 'On Loan', display a message and roll back the deletion

RAISERROR ('Artwork cannot be deleted because it is either Sold or On Loan.', 16, 1);

ROLLBACK TRANSACTION;

END

ELSE

BEGIN

-- If none of the deleted artworks are 'Sold' or 'On Loan', proceed with the deletion

DELETE FROM Artwork WHERE ArtID IN (SELECT ArtID FROM deleted);

END

END;

CREATE TRIGGER trg\_DeleteArtworkReferences

ON Artwork

AFTER DELETE

AS

BEGIN

DECLARE @ArtID INT;

SELECT @ArtID = ArtID FROM deleted;

DELETE FROM Sales WHERE ArtID = @ArtID;

DELETE FROM Loan WHERE ArtID = @ArtID;

DELETE FROM Exhibition WHERE ArtID = @ArtID;

END;

GO

CREATE TRIGGER trg\_DeleteArtistReferences

ON Artist

AFTER DELETE

AS

BEGIN

DECLARE @AID INT;

SELECT @AID = AID FROM deleted;

DELETE FROM Artwork WHERE AID = @AID;

DELETE FROM Sales WHERE ArtID IN (SELECT ArtID FROM Artwork WHERE AID = @AID);

DELETE FROM Loan WHERE ArtID IN (SELECT ArtID FROM Artwork WHERE AID = @AID);

DELETE FROM Exhibition WHERE ArtID IN (SELECT ArtID FROM Artwork WHERE AID = @AID);

END;

GO

CREATE TRIGGER trg\_DeleteStaffUserReferences

ON Users

AFTER DELETE

AS

BEGIN

DECLARE @UserID INT;

DECLARE @Role VARCHAR(100);

SELECT @UserID = UserID, @Role = Role FROM deleted;

IF @Role = 'staff'

BEGIN

DELETE FROM Sales WHERE UserID = @UserID;

DELETE FROM Loan WHERE UserID = @UserID;

END

END;

GO

-- Trigger to delete dependent records in Sales, Loan, and Gallery tables when a User is deleted

CREATE TRIGGER trg\_DeleteDependentRecordsOnUserDelete

ON Users

INSTEAD OF DELETE

AS

BEGIN

DELETE FROM Sales WHERE UserID IN (SELECT UserID FROM deleted);

DELETE FROM Loan WHERE UserID IN (SELECT UserID FROM deleted);

DELETE FROM Gallery WHERE UserID IN (SELECT UserID FROM deleted);

DELETE FROM Users WHERE UserID IN (SELECT UserID FROM deleted);

END;

GO

-- Trigger to delete dependent records in Exhibition table when an Artwork is deleted

CREATE TRIGGER trg\_DeleteDependentRecordsOnArtworkDelete

ON Artwork

AFTER DELETE

AS

BEGIN

DELETE FROM Exhibition WHERE ArtID IN (SELECT ArtID FROM deleted);

DELETE FROM Sales WHERE ArtID IN (SELECT ArtID FROM deleted);

DELETE FROM Loan WHERE ArtID IN (SELECT ArtID FROM deleted);

END;

GO

-- Trigger to delete dependent records in Artwork and other tables when an Artist is deleted

CREATE TRIGGER trg\_DeleteDependentRecordsOnArtistDelete

ON Artist

AFTER DELETE

AS

BEGIN

DELETE FROM Artwork WHERE AID IN (SELECT AID FROM deleted);

DELETE FROM Exhibition WHERE ArtID IN (SELECT ArtID FROM Artwork WHERE AID IN (SELECT AID FROM deleted));

DELETE FROM Sales WHERE ArtID IN (SELECT ArtID FROM Artwork WHERE AID IN (SELECT AID FROM deleted));

DELETE FROM Loan WHERE ArtID IN (SELECT ArtID FROM Artwork WHERE AID IN (SELECT AID FROM deleted));

DELETE FROM Artist WHERE AID IN (SELECT AID FROM deleted);

END;

GO

**Question 3 :**

**History Table**

CREATE VIEW ClientPurchaseHistoryReport AS

SELECT

C.UserID AS ClientID,

CONCAT(U.FirstName, ' ', U.LastName) AS ClientName,

P.ArtID,

W.Title AS ArtworkTitle,

P.SaleDate AS PurchaseDate

FROM

Users U

JOIN

Sales P ON U.UserID = P.UserID

JOIN

Artwork W ON P.ArtID = W.ArtID

JOIN

Artist A ON W.AID = A.AID

JOIN

Users C ON A.UserID = C.UserID;

GO

****