**ARINDA HILLARY**

**DBMS**

**PROJECT: PART 2**

**3/19/2024**

**﻿STORED PROCEDURES**

**﻿1) Write a stored procedure that takes in one argument, the staff number of an instructor.**

**The procedure outputs all details of all the lessons for that instructor**

CREATE OR ALTER PROCEDURE GetInstructorLessons

@InstructorID INT

AS

BEGIN

SELECT \*

FROM Lesson

WHERE staffID = @InstructorID;

END;

EXEC GetInstructorLessons @InstructorID = 22

﻿**2) Write a stored procedure that takes in two arguments, a staff number and a date. The**

**procedure shows details of all lessons for that staff instructor, starting at the date of the**

**argument, and ending seven days later**

CREATE OR ALTER PROCEDURE GetInstructorLessonsWithinWeek

@InstructorID INT,

@StartDate DATE

AS

BEGIN

SELECT \*

FROM Lesson

WHERE staffID = @InstructorID

AND lessonDate BETWEEN @StartDate AND DATEADD(day, 7, @StartDate);

END;

EXEC GetInstructorLessonsWithinWeek @InstructorID = 22, @StartDate = '2023-4-1'

**﻿3) Do the same as questions 1 and 2 above, but for a client number instead of a staff**

**number.**

CREATE OR ALTER PROCEDURE GetClientLessons

@ClientID INT

AS

BEGIN

SELECT \*

FROM Lesson

WHERE clientID = @ClientID;

END;

EXEC GetClientLessons @ClientID = 1;

CREATE OR ALTER PROCEDURE GetClientLessonsWithinWeek

@ClientID INT,

@StartDate DATE

AS

BEGIN

SELECT \*

FROM Lesson

WHERE clientID = @ClientID

AND lessonDate BETWEEN @StartDate AND DATEADD(day, 6, @StartDate);

END;

EXEC GetClientLessonsWithinWeek @ClientID = 1, @StartDate ='2023-1-27';

**﻿4) Create some stored procedures yourself which do something you would like to see being**

**done.**

**Create a stored procedure that retrieves all available instructors (Senior Instructor and Instructor) who do not have any scheduled lessons on a given date.**

CREATE OR ALTER PROCEDURE GetAvailableInstructors

@SpecificDate DATE

AS

BEGIN

SELECT

staffID,

fName,

lName

FROM

Staff

WHERE

(position = 'Senior Instructor' OR position = 'Instructor')

AND staffID NOT IN (

SELECT staffID

FROM Lesson

WHERE lessonDate = @SpecificDate

);

END;

EXEC GetAvailableInstructors @SpecificDate='2023-1-30'

**VIEWS**

**﻿5) Create a View called Client\_Lesson which does an inner join on the Client and Lesson**

**tables. Run it to make sure it works properly!**

CREATE VIEW Client\_Lesson AS

SELECT

Client.clientID,

Client.fName,

Client.lName,

Client.dob,

Client.phone,

Client.city,

Client.zipCode,

Lesson.lessonID,

Lesson.lessonDate,

Lesson.topicCovered,

Lesson.staffID,

Lesson.mileage,

Lesson.instructorComments,

Lesson.lessonStatus,

Lesson.startTime

FROM

Client

INNER JOIN

Lesson

ON Client.clientID = Lesson.clientID;

GO

SELECT \* FROM Client\_Lesson;

﻿**6) Create a View called Lesson\_Info which calls the View above Client\_Lesson, and outputs**

**all the information from Client\_Lesson, along with who the staff person is for the lesson,**

**i.e. the staff p﻿erson’s name and staffID.**

CREATE VIEW Lesson\_Info AS

SELECT

cl.\*,

Staff.fName AS StaffFirstName,

Staff.lName AS StaffLastName

FROM

Client\_Lesson cl

INNER JOIN

Staff

ON cl.staffID = Staff.staffID;

**﻿7) Create two more views that may be useful to you. Test them!**

1. **View for Instructors with Upcoming Birthdays**

CREATE VIEW UpcomingInstructorBirthdays AS

SELECT

staffID,

fName,

lName,

dob

FROM

Staff

WHERE

MONTH(dob) = MONTH(GETDATE()) AND DAY(dob) >= DAY(GETDATE())

OR MONTH(dob) = MONTH(DATEADD(MONTH, 1, GETDATE())) AND DAY(dob) <= DAY(GETDATE());

GO

SELECT \* from UpcomingInstructorBirthdays;

1. **View for Upcoming Driving Tests**

CREATE VIEW UpcomingDrivingTests AS

SELECT

testID,

clientID,

testDate,

testTime

FROM

DrivingTest

WHERE

testDate BETWEEN GETDATE() AND DATEADD(DAY, 30, GETDATE());

GO

SELECT \* FROM UpcomingDrivingTests;

**USER DEFINED FUNCTIONS**

**﻿8) Create a user defined function that returns the total lessons that a client has taken up**

**to today.**

CREATE OR ALTER FUNCTION GetTotalLessons(@ClientID INT)

RETURNS INT

AS

BEGIN

DECLARE @LessonCount INT;

SELECT @LessonCount = COUNT(\*)

FROM Lesson

WHERE clientID = @ClientID AND lessonDate <= GETDATE();

RETURN @LessonCount;

END;

GO

SELECT dbo.GetTotalLessons( 2) AS 'TOTAL LESSONS'

**9) Create a user defined function that returns the total lessons that a client has taken**

**before a date supplied by the user.**

CREATE OR ALTER FUNCTION GetLessonsBeforeDate(@ClientID INT, @BeforeDate DATE)

RETURNS INT

AS

BEGIN

DECLARE @LessonCount INT;

SELECT @LessonCount = COUNT(\*)

FROM Lesson

WHERE clientID = @ClientID AND lessonDate < @BeforeDate;

RETURN @LessonCount;

END;

GO

SELECT dbo.GetLessonsBeforeDate( 1, '2023-4-1') AS 'lessons before given date'

**10)Create a user defined function that returns a table which does an inner join on the Client**

**and Lesson tables, for a particular client which is supplied by the user. Run it to make**

**sure it works properly!**

CREATE OR ALTER FUNCTION GetLessonDetails(@ClientID INT)

RETURNS TABLE

AS

RETURN (

SELECT

Client.clientID,

Client.fName,

Client.lName,

Client.dob,

Client.phone,

Client.city,

Client.zipCode,

Lesson.lessonID,

Lesson.lessonDate,

Lesson.topicCovered,

Lesson.staffID,

Lesson.mileage,

Lesson.instructorComments,

Lesson.lessonStatus,

Lesson.startTime

FROM

Lesson

INNER JOIN

Client ON Lesson.clientID =Client.clientID

WHERE

Client.clientID = @ClientID

);

GO

SELECT \* FROM dbo.GetLessonDetails (1);

**TRIGGERS**

--Alter table to add the column

ALTER TABLE Staff

ADD totalClients INT DEFAULT 0;

GO

UPDATE Staff

SET totalClients = (SELECT COUNT(DISTINCT clientID) FROM Lesson WHERE staffID = Staff.staffID);

GO

--trigger for incrementing

CREATE TRIGGER trg\_IncrementTotalClients

ON Lesson

AFTER INSERT

AS

BEGIN

UPDATE Staff

SET totalClients = totalClients + 1

WHERE staffID IN (SELECT DISTINCT staffID FROM inserted);

END;

GO

--trigger for decrementing

CREATE TRIGGER trg\_DecrementTotalClients

ON Lesson

AFTER DELETE

AS

BEGIN

UPDATE Staff

SET totalClients = totalClients - 1

WHERE staffID IN (SELECT DISTINCT staffID FROM deleted);

END;

GO

**CURSORS**