EXAMEN 03-06-2015 JUNIO BASE DE DATOS HECTOR GARCÍA GONZÁLEZ

**1)Crea una función tal que liste todos los clientes que ahyan estado o estén alojados en un hotel que se le pasa. Utiliza la función para el hotel CARLTON**

USE JunioProfe

GO

CREATE FUNCTION DBO.ListarCLIENTES (@hotelNOMBRE NVARCHAR(15))

RETURNS TABLE

AS

RETURN --REALIZARLO CON SUBCONSULTAS PARA QUE NO SE REPITAN CLIENTES--

(

SELECT c.codcli, c.nombre, c.apellido, c.ciucli, c.dni FROM cliente c WHERE c.codcli IN

(SELECT e.codcli FROM estancia e WHERE e.codcli = c.codcli AND e.fechaentrada IS NOT NULL AND e.codhot IN

( SELECT hotel.codhot from hotel WHERE hotel.nombre = @hotelNOMBRE )

))

GO

SELECT \* FROM DBO.ListarCLIENTES('CARLTON')

GO

**2)Crea una función para incrementar el precio que se le pasa en un 21% de IVA. Utiliza dicha función para sacar por pantalla el nombre del hotel, tipo de habitación, precio anterior y el precio con el IVA aplicado**

USE JunioProfe

GO

CREATE FUNCTION DBO.IncrementarPrecio (@IVA INT)

RETURNS TABLE

AS

RETURN

(

SELECT h.nombre, hab.tipohab, hab.precionoche as PRECIO\_ANTERIOR, hab.precionoche\*@IVA as PRECIO\_CON\_IVA\_APLICADO

FROM hotel h , habitacion hab

)

GO

SELECT \* FROM DBO.IncrementarPrecio(21)

GO

PARECIDO AL EJERCICIO DE FUNCIONES QUE CREABAS UNA FUNCION

Y LUEGO LA LLAMABAS JUNTO A OTRAS

USE JunioProfe

GO

CREATE FUNCTION DBO.IncrementarPrecio (@IVA INT)

RETURNS DECIMAL(9,2)

AS

BEGIN

DECLARE @Resultado decimal(9,2)

SET @Resultado = @Cantidad \*1.21

RETURN(@Resultado)

END

GO

SELECT hotel.nombre, DBO.IncrementarPrecio(habitacion.precionoche) as PRECIOnocheACTUALIZADO from hotel

**3)Realiza lo necesario para que cuando se cierre una estancia (un cliente abandone una habitacion), la habitación que ocupadba pase a estar automáticamente desocupada**

USE JunioProfe

GO

CREATE TRIGGER abandonarHabitacionquitarESTANCIA1122 on habitacion

AFTER UPDATE

AS

BEGIN

IF UPDATE(fechasalida)

BEGIN

UPDATE habitacion SET ocupada = 'NO'

WHERE habitacion.numhab IN(SELECT hotel.codhot FROM hotel WHERE hotel.codhot IN (SELECT estancia.codhot FROM estancia

WHERE estancia.fechasalida IS NOT NULL))

print 'aviso, hay una habitacion desocupada'

END

END

GO

UPDATE estancia SET fechasalida = GETDATE() WHERE codhot = 2

GO

USE JunioProfe

GO

CREATE TRIGGER abandonarHabitacionquitarESTANCIA1122 on habitacion

INSTEAD OF UPDATE AS

RETURN

BEGIN

IF @@ROWCOUNT = 0

BEGIN

UPDATE habitacion SET ocupada = 'NO'

WHERE habitacion.numhab IN(SELECT hotel.codhot FROM hotel WHERE hotel.codhot IN (SELECT estancia.codhot FROM estancia

WHERE estancia.fechasalida IS NOT NULL))

print 'aviso, hay una habitacion desocupada'

END

END

FORMA CORRECTA

USE JunioProfe

GO

CREATE TRIGGER abandonarHabitacionquitarESTANCIA1122 on habitacion

AFTER UPDATE

AS

BEGIN

IF UPDATE(fechasalida)

BEGIN

UPDATE habitacion SET ocupada = 'NO'

WHERE habitacion.numhab IN(SELECT hotel.codhot FROM hotel WHERE hotel.codhot IN (SELECT estancia.codhot FROM estancia

WHERE estancia.fechasalida IS NOT NULL))

print 'aviso, hay una habitacion desocupada'

END

END

GO

UPDATE estancia SET fechasalida = GETDATE() WHERE codhot = 2

GO

**4)Crea un procedimiento almacenado al que se pase como parámetro dni y ciudad y nos devuelva el numero de veces que se ha alojado en hoteles de esa ciudad. Verifica que se pasan los datos.**

USE JunioProfe

GO

CREATE PROCEDURE numeroVECESclientes11

@DNI nvarchar(9) = '16511231r',

@CIUDAD varchar(20) = 'Logroño'

AS

SELECT COUNT(e.idestancia) AS Numero\_Estancias FROM estancia e

JOIN hotel h ON h.codhot = e.codhot AND h.ciuhot = @CIUDAD

JOIN cliente c ON c.codcli = e.codcli AND c.dni = @DNI

GO

EXEC numeroVECESclientes11

GO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CORRECCION CORRECTA

USE JunioProfe

GO

CREATE PROCEDURE numeroVECESclientes11

@DNI nvarchar(9) ,

@CIUDAD varchar(20),

@NUMERO INT OUTPUT

AS

SET @NUMERO = 0

IF @DNI IS NULL OR @CIUDAD IS NULL

RAISERROR ('PASO DE PARAMETROS ERRONEOS',14,1)

ELSE

SELECT @NUMERO=COUNT(\*) FROM estancia e

JOIN hotel h ON h.codhot = e.codhot AND h.ciuhot = @CIUDAD

JOIN cliente c ON c.codcli = e.codcli AND c.dni = @DNI

RETURN(@NUMERO)

GO

DECLARE @NUMERO INT

GO

EXEC numeroVECESclientes11 '16511231r','Logroño',@NUMERO OUTPUT

SELECT @NUMERO

GO

**5) No permitir que se realiza una nueva estancia de un cliente que permanece alojado todavía en alguin hotel**

USE JunioProfe

go

CREATE TRIGGER NOpermitirnuevaESTANCIA

ON estancia

AFTER INSERT AS

BEGIN

--si ya estoy metido en un hotel no le puedo permitir una nueva estancia

IF ( SELECT count(e.numhab) FROM habitacion AS e, inserted i WHERE e.numhab = i.numhab AND e.codhot = i.codhot

AND UPPER(e.ocupada)='SI') > 0 --- SI ES if exist no se puede poner luego > 0

BEGIN

PRINT 'NO SE PERMITE NUEVA ESTANCIA, ELIJA OTRA'

ROLLBACK TRANSACTION

END

--si la habitacion esta ocupada no le puedo permitir entrar en esa habitacion

IF(SELECT COUNT(\*) FROM inserted i, estancia e WHERE e.codcli = i.codcli AND e.fechasalida IS NULL) >= 2

BEGIN

PRINT 'NO SE PERMITE NUEVA ESTANCIA, HABITACION OCUPADA'

ROLLBACK TRANSACTION

END

END

GO

INSERT estancia

VALUES (1,1,1,104,GETDATE(), NULL)

6) Crea una función que pasándole el dni del cliente te diga el numero de veces uqe ha estado (incluyendo si actualmente esta) en cada hotel, mostrando por pantalla el nombre del hotel, numero de veces.

USE JunioProfe

GO

CREATE FUNCTION numeroVECESclientesESTA1 (@DNI NVARCHAR(9))

RETURNS TABLE

AS

RETURN

(

SELECT COUNT(e.idestancia) AS Numero\_Estancias, h.nombre FROM estancia e

JOIN hotel h ON h.codhot = e.codhot

JOIN cliente c ON c.codcli = e.codcli AND c.dni = @DNI

WHERE e.fechaentrada IS NOT NULL --AND e.fechasalida IS NULL

GROUP BY h.nombre

)

GO

SELECT \* FROM numeroVECESclientesESTA1('16511231r')

GO

**7)Usando cursores actualiza el precio de las habitaciones sin es superior, se descuenta el precio de la habitación al 20 por ciento si no al 50. Para cualquier hoteñ**

USE JunioProfe

GO

DECLARE @numhab SMALLINT,@codhot SMALLINT, @SUPERFICIE INT,@tipohab NVARCHAR(10),@precionoche MONEY,@ocupada NVARCHAR(2)

DECLARE PrecioHabitaciones CURSOR FOR

SELECT h.numhab, h.codhot, h.superficie, h.tipohab, h.precionoche, h.ocupada FROM habitacion h FOR UPDATE

OPEN PrecioHabitaciones

FETCH PrecioHabitaciones INTO @numhab, @codhot, @SUPERFICIE, @tipohab,@precionoche,@ocupada

WHILE (@@FETCH\_STATUS = 0 )

BEGIN

IF (@tipohab = 'Superior')

SET @precionoche = @precionoche - (@precionoche/100)\*20

ELSE

SET @precionoche = @precionoche - (@precionoche/100)\*50

UPDATE habitacion SET precionoche = @precionoche WHERE CURRENT OF PrecioHabitaciones

FETCH PrecioHabitaciones INTO @numhab, @codhot, @SUPERFICIE, @tipohab,@precionoche,@ocupada

END

CLOSE PrecioHabitaciones

DEALLOCATE PrecioHabitaciones

**8) En el proceso de restauración de una base de datos cita y describe los estados en los que puede estar la base de datos.**

Stand by

Recovery

Non recovery

Cuando hacemos una copia de seguridad de la BD puede ser completa o diferencial. La completa se realiza la primera vez que hacemos la copia y cuando realizamos copias de gran tamaño pero que han sufrido pequeños cambios. La diferencial la hacemos cuando es más habitualmente modificarla ya que la diferencial solo copia las modificaciones sufridas desde la ultima copia.

Cuando restauramos debemos elegir cual de esas copias queremos restaurar

**9)Realiza mediante comandos, una copia completa y diferencial de la junioprofe a la carpeta por defecto.**

BACKUP DATABASE [Junioprofe]

TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLDAW110\MSSQL\Backup\junioprofe. back'

--completa

GO

BACKUP DATABASE [Junioprofe]

TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLDAW110\MSSQL\Backup\junioprofe. back'

WITH DIFFERENTIAL--para señalar copia diferencial

GO