```sql
CREATE DATABASE coco_tours_db ON PRIMARY
( NAME = 'cocotours_db_data',
 FILENAME ="C:\data\cocotours_db_data.mdf",
 SIZE = 100MB ,
 MAXSIZE = 150MB,
 FILEGROWTH = 5MB
)
LOG ON
(NAME = 'cocotours_db_log',
 FILENAME = "C:\log\cocotours_db_log.ldf" ,
 SIZE = 5MB ,
 MAXSIZE = 50MB,
 FILEGROWTH = 5%
)
COLLATE Modern_Spanish_CI_AI
GO

-- Use database
USE coco_tours_db;
GO

-- Table: client
CREATE TABLE client (
    id INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);
GO

-- Table: client_phones
CREATE TABLE client_phones (
    id INT IDENTITY(1,1) PRIMARY KEY,
    client_id INT NOT NULL,
    phone VARCHAR(20) NOT NULL,
    FOREIGN KEY (client_id) REFERENCES client(id) ON DELETE CASCADE
);
GO

-- Table: administrator
CREATE TABLE administrator (
    id INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);
GO

-- Table: user
CREATE TABLE [user] (
    id INT IDENTITY(1,1) PRIMARY KEY,
    email VARCHAR(70) UNIQUE NOT NULL,
    username VARCHAR(30) UNIQUE NOT NULL,
    password VARCHAR(150) NOT NULL,
    client_id INT DEFAULT NULL,
    admin_id INT DEFAULT NULL,
    FOREIGN KEY (client_id) REFERENCES client(id) ON DELETE SET NULL,
    FOREIGN KEY (admin_id) REFERENCES administrator(id) ON DELETE SET
NULL
    );
GO

-- Table: extra
CREATE TABLE extra (
    id INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(40) NOT NULL,
    description VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL
);
GO

-- Table: reservations
CREATE TABLE reservations (
    id INT IDENTITY(1,1) PRIMARY KEY,
    date DATE NOT NULL,
    time TIME NOT NULL,
    description VARCHAR(100) NOT NULL,
    tour_subtotal DECIMAL(10,2) DEFAULT 0,
    extra_subtotal DECIMAL(10,2) DEFAULT 0,
    total DECIMAL(10,2) DEFAULT 0,
    [user_id] INT NOT NULL,
    FOREIGN KEY ([user_id]) REFERENCES [user](id) ON DELETE CASCADE
);
GO

-- Table: extra_detail
CREATE TABLE extra_detail (
    id INT IDENTITY(1,1) PRIMARY KEY,
    person_count INT NOT NULL,
    total_price DECIMAL(10,2),
    extra_id INT NOT NULL,
    reservation_id INT NOT NULL,
    FOREIGN KEY (reservation_id) REFERENCES reservations(id) ON DELETE
CASCADE,
    FOREIGN KEY (extra_id) REFERENCES extra(id) ON DELETE CASCADE
);
GO

-- Table: supplier
CREATE TABLE supplier (
    id INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description VARCHAR(100) NOT NULL,
    email VARCHAR(70) NOT NULL
);
GO

-- Table: supplier_phones
CREATE TABLE supplier_phones (
    id INT IDENTITY(1,1) PRIMARY KEY,
    phone VARCHAR(20) NOT NULL,
    supplier_id INT NOT NULL,
    FOREIGN KEY (supplier_id) REFERENCES supplier(id) ON DELETE CASCADE
);
GO

-- Table: tour
CREATE TABLE tour (
    id INT IDENTITY(1,1) PRIMARY KEY,
    type VARCHAR(30) NOT NULL,
    description VARCHAR(MAX) NOT NULL,
    price DECIMAL(10,2) NOT NULL
);
GO

-- Table: tour_detail
CREATE TABLE tour_detail (
    id INT IDENTITY(1,1) PRIMARY KEY,
    passenger_count INT NOT NULL,
    origin VARCHAR(40) NOT NULL,
    destination VARCHAR(40) NOT NULL,
    tour_id INT NOT NULL,
    reservation_id INT NOT NULL,
    supplier_id INT NOT NULL,
    FOREIGN KEY (tour_id) REFERENCES tour(id) ON DELETE CASCADE,
    FOREIGN KEY (reservation_id) REFERENCES reservations(id) ON DELETE
CASCADE,
    FOREIGN KEY (supplier_id) REFERENCES supplier(id) ON DELETE CASCADE
);
GO

-- Table: passengers
CREATE TABLE passengers (
    id INT IDENTITY(1,1) PRIMARY KEY,
    name VARCHAR(60) NOT NULL,
    age INT NOT NULL,
    tour_detail_id INT NOT NULL,
    FOREIGN KEY (tour_detail_id) REFERENCES tour_detail(id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
GO

-- Table: users audting
CREATE TABLE auditing_users_management(
 id INT IDENTITY(1,1) PRIMARY KEY,
 table_name VARCHAR(30),
 type_action VARCHAR(15),
 record_id INT,
 username VARCHAR(70) NULL,
 changed_by VARCHAR(70) NULL,
 change_date DATETIME DEFAULT GETDATE(),
 old_data NVARCHAR(MAX) NULL,
 new_data NVARCHAR(MAX) NULL
);
GO
```

```
-- Table: operation auditing
CREATE TABLE auditing_operation_management(
  id INT IDENTITY(1,1) PRIMARY KEY,
 table_name VARCHAR(30),
 type_action VARCHAR(15),
 record_id INT,
 username VARCHAR(70) NULL,
 changed_by VARCHAR(70) NULL,
 change_date DATETIME DEFAULT GETDATE(),
 old_data NVARCHAR(MAX) NULL,
 new_data NVARCHAR(MAX) NULL
);
GO


-- Procedimientos almacenados y triggers

--Trigger

CREATE TRIGGER trg_set_user_role
ON [user]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE u
    SET role =
        CASE
            WHEN i.client_id IS NOT NULL THEN 'CLIENT'
            WHEN i.admin_id IS NOT NULL THEN 'ADMIN'
            ELSE NULL
        END
    FROM [user] u
    INNER JOIN inserted i ON u.id = i.id;
END;

--

CREATE OR ALTER TRIGGER trg_tour_detail_update
ON tour_detail
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @reservation_id INT;

    DECLARE res_cursor CURSOR LOCAL FAST_FORWARD FOR
        SELECT DISTINCT reservation_id FROM inserted
        UNION
        SELECT DISTINCT reservation_id FROM deleted;
    OPEN res_cursor;
    FETCH NEXT FROM res_cursor INTO @reservation_id;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        DECLARE @tour_subtotal DECIMAL(10,2);

        SELECT @tour_subtotal = ISNULL(SUM(t.price),0)
        FROM tour_detail td
        JOIN tour t ON td.tour_id = t.id
        WHERE td.reservation_id = @reservation_id;

        UPDATE reservations
        SET tour_subtotal = @tour_subtotal,
            total = @tour_subtotal + ISNULL(extra_subtotal,0)
        WHERE id = @reservation_id;

        FETCH NEXT FROM res_cursor INTO @reservation_id;
    END

    CLOSE res_cursor;
    DEALLOCATE res_cursor;
END;
GO

--

CREATE OR ALTER TRIGGER trg_extra_detail_update
ON extra_detail
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE ed
    SET ed.total_price = ed.person_count * e.unit_price
    FROM extra_detail ed
    JOIN extra e ON ed.extra_id = e.id
    JOIN inserted i ON ed.id = i.id

    DECLARE @reservation_id INT;


    DECLARE res_cursor CURSOR LOCAL FAST_FORWARD FOR
        SELECT DISTINCT reservation_id FROM inserted
        UNION
        SELECT DISTINCT reservation_id FROM deleted;

    OPEN res_cursor;
    FETCH NEXT FROM res_cursor INTO @reservation_id;
    WHILE @@FETCH_STATUS = 0
    BEGIN
        DECLARE @extra_subtotal DECIMAL(10,2);
        SELECT @extra_subtotal = ISNULL(SUM(total_price),0)
        FROM extra_detail
        WHERE reservation_id = @reservation_id;
        UPDATE reservations
        SET extra_subtotal = @extra_subtotal,
            total = @extra_subtotal + ISNULL(tour_subtotal,0)
        WHERE id = @reservation_id;

        FETCH NEXT FROM res_cursor INTO @reservation_id;
    END

    CLOSE res_cursor;
    DEALLOCATE res_cursor;
END;
GO

--

USE coco_tours_db
GO

CREATE TRIGGER trg_audit_admin
ON administrator
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
  INSERT INTO
auditing_users_management(table_name,type_action,record_id,old_data,new_d
ata)
  SELECT
    'administrator',
     CASE
            WHEN i.id IS NOT NULL AND d.id IS NOT NULL THEN 'UPDATE'
            WHEN i.id IS NOT NULL THEN 'INSERT'
            ELSE 'DELETE'
     END,
     COALESCE (I.id, D.id),
     d.name,
     i.name
  FROM inserted AS I
  FULL JOIN deleted AS D ON I.id = D.id;
END;
GO

--

USE coco_tours_db
GO

CREATE TRIGGER trg_audit_client
ON client
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
```

```
  INSERT INTO
auditing_users_management(table_name,type_action,record_id,old_data,new_d
ata)
  SELECT
    'client',
     CASE
            WHEN i.id IS NOT NULL AND d.id IS NOT NULL THEN 'UPDATE'
            WHEN i.id IS NOT NULL THEN 'INSERT'
            ELSE 'DELETE'
     END,
     COALESCE (I.id, D.id),
     d.name,
     i.name
  FROM inserted AS I
  FULL JOIN deleted AS D ON I.id = D.id;
END;
GO

--

USE coco_tours_db
GO

CREATE TRIGGER trg_audit_extra_detail
ON extra_detail
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
  INSERT INTO
auditing_operation_management(table_name,type_action,record_id,old_data,n
ew_data)
  SELECT
    'extra_detail',
     CASE
            WHEN i.id IS NOT NULL AND d.id IS NOT NULL THEN 'UPDATE'
            WHEN i.id IS NOT NULL THEN 'INSERT'
            ELSE 'DELETE'
     END,
     COALESCE (I.id, D.id),
     CAST(D.total_price AS nvarchar),
     CAST(I.total_price AS nvarchar)
  FROM inserted AS I
  FULL JOIN deleted AS D ON I.id = D.id;
END;
GO

--

USE coco_tours_db
GO

CREATE TRIGGER trg_audit_reservations
ON reservations
AFTER INSERT, UPDATE, DELETE
```

```
AS
BEGIN
  INSERT INTO
auditing_operation_management(table_name,type_action,record_id,old_data,n
ew_data)
  SELECT
    'reservations',
     CASE
            WHEN i.id IS NOT NULL AND d.id IS NOT NULL THEN 'UPDATE'
            WHEN i.id IS NOT NULL THEN 'INSERT'
            ELSE 'DELETE'
     END,
     COALESCE (I.id, D.id),
     CAST(D.total AS nvarchar),
     CAST(I.total AS nvarchar)
  FROM inserted AS I
  FULL JOIN deleted AS D ON I.id = D.id;
END;
GO

--

USE coco_tours_db
GO

CREATE TRIGGER trg_audit_tour_detail
ON tour_detail
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
  INSERT INTO
auditing_operation_management(table_name,type_action,record_id,old_data,n
ew_data)
  SELECT
    'tour_detail',
     CASE
            WHEN i.id IS NOT NULL AND d.id IS NOT NULL THEN 'UPDATE'
            WHEN i.id IS NOT NULL THEN 'INSERT'
            ELSE 'DELETE'
     END,
     COALESCE (I.id, D.id),
     CAST(D.tour_id AS nvarchar),
     CAST(I.tour_id AS nvarchar)
  FROM inserted AS I
  FULL JOIN deleted AS D ON I.id = D.id;
END;
GO

--
USE coco_tours_db
GO

CREATE TRIGGER trg_audit_user
ON [user]
```

```
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
  INSERT INTO
auditing_users_management(table_name,type_action,record_id,old_data,new_d
ata)
  SELECT
    'user',
     CASE
            WHEN i.id IS NOT NULL AND d.id IS NOT NULL THEN 'UPDATE'
            WHEN i.id IS NOT NULL THEN 'INSERT'
            ELSE 'DELETE'
     END,
     COALESCE (I.id, D.id),
     d.username,
     i.username
  FROM inserted AS I
  FULL JOIN deleted AS D ON I.id = D.id;
END;
GO

--


---PA

CREATE PROCEDURE pa_admin_delete(
 @user_id INT
)AS
BEGIN
BEGIN TRY


 IF NOT EXISTS(SELECT 1 FROM [user] WHERE id = @user_id AND [role] =
'ADMIN')
 BEGIN
  RAISERROR('El usuario administrador no existe', 16, 1)
  RETURN
 END
 --
 BEGIN TRANSACTION
 DECLARE @admin_id INT
 SELECT @admin_id = a.id FROM administrator AS a INNER JOIN [user] AS u
ON a.id = u.admin_id
           WHERE u.id = @user_id
DELETE FROM [user] WHERE id = @user_id
DELETE FROM administrator WHERE id = @admin_id
 COMMIT TRANSACTION
 --

END TRY
BEGIN CATCH
 DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE()
 RAISERROR(@ErrorMessage, 16, 1)
 RETURN
END CATCH
END
GO

CREATE PROCEDURE pa_admin_insert (
      @name VARCHAR(50),
      @email VARCHAR(70),
      @username VARCHAR(30),
      @password VARCHAR(150)
) AS
BEGIN
BEGIN TRY


 IF @name IS NULL OR LTRIM(RTRIM(@name)) = ''
 BEGIN
  RAISERROR('El nombre no puede ser un valor vacio', 16, 1)
  RETURN
 END

IF @email NOT LIKE '%_@_%._%'
BEGIN
 RAISERROR('El formato email no es válido', 16, 1)
 RETURN
END
ELSE IF exists (SELECT 1 FROM [user] WHERE email = @email)
BEGIN
 RAISERROR('El correo ya esta asosiado a un usuario', 16, 1)
 RETURN
END

IF @username IS NULL OR LTRIM(RTRIM(@username)) = ''
BEGIN
 RAISERROR('El nombre de usuario  no puede ser un valor vacio', 16, 1)
 RETURN;
END
ELSE IF exists (SELECT 1 FROM [user] WHERE username = @username)
BEGIN
 RAISERROR('El nombre de usuario ya se encuentra en uso', 16, 1)
 RETURN
END

IF @password IS NULL OR LTRIM(RTRIM(@password)) = ''
BEGIN
RAISERROR('la contraseña no es valida', 16, 1)
 RETURN
END
ELSE IF (@password NOT LIKE '%[0-9]%' AND @password NOT LIKE
'%[!@#$^&*()]%')
BEGIN
 RAISERROR('La contraseña debe incluir un numero y un caracters
especial', 16, 1)
 RETURN
```

```
END

--
INSERT INTO administrator ([name]) VALUES (@name)

DECLARE @admin_id INT = SCOPE_IDENTITY()

INSERT INTO [user](email, username, [password], admin_id, client_id)
VALUES(@email,@username,@password,@admin_id,NULL)

END  TRY
BEGIN CATCH
      RAISERROR('Hubo un error en el proceso, por favor, revise los datos
ingresados', 16, 1)
      RETURN
END CATCH
END
GO

CREATE PROCEDURE pa_admin_update (
    @user_id INT,
    @new_name VARCHAR(50),
    @new_email VARCHAR(70),
    @new_username VARCHAR(30),
    @new_password VARCHAR(150)
) AS
BEGIN
BEGIN TRY
DECLARE @admin_id INT
SELECT @admin_id = a.id FROM administrator AS a INNER JOIN [user] AS u ON
a.id = u.admin_id
WHERE u.id = @user_id
 IF NOT EXISTS (SELECT 1 FROM [user] WHERE id = @user_id)
 BEGIN
  RAISERROR('El administrador no existe', 16, 1)
  RETURN
 END

 IF LTRIM(RTRIM(@new_name)) = ''
 BEGIN
  RAISERROR('El nombre no puede ser un valor vacio', 16, 1)
  RETURN
 END

 IF @new_email IS NOT NULL
 BEGIN
 IF @new_email NOT LIKE '%_@_%._%'
 BEGIN
  RAISERROR('El formato email no es válido', 16, 1)
  RETURN
 END
 ELSE IF EXISTS (SELECT 1 FROM [user] WHERE email = @new_email AND
admin_id <> @admin_id)
 BEGIN
```

```
  RAISERROR('El correo ya está asociado a otro usuario', 16, 1)
  RETURN
 END
 END

 IF @new_username IS NOT NULL
 BEGIN
 IF LTRIM(RTRIM(@new_username)) = ''
 BEGIN
  RAISERROR('El nombre de usuario no puede ser un valor vacio', 16, 1)
  RETURN
 END
 ELSE IF EXISTS (SELECT 1 FROM [user] WHERE username = @new_username AND
admin_id <> @admin_id)
 BEGIN
  RAISERROR('El nombre de usuario ya se encuentra en uso', 16, 1)
  RETURN
 END
 END

 IF @new_password IS NOT NULL
 BEGIN
 IF LTRIM(RTRIM(@new_password)) = ''
 BEGIN
  RAISERROR('La contraseña  no puede ser un valor vacio', 16, 1)
  RETURN
 END
 ELSE IF (@new_password NOT LIKE '%[0-9]%' AND @new_password NOT LIKE
'%[!@#$^&*()]%')
 BEGIN
  RAISERROR('La contraseña debe incluir un número y un carácter
especial', 16, 1)
  RETURN
 END
 END

   UPDATE administrator SET name = ISNULL(@new_name, name) WHERE id =
@admin_id;

    UPDATE [user]
    SET email = ISNULL(@new_email, email),
        username = ISNULL(@new_username, username),
        [password] = ISNULL(@new_password, [password])
    WHERE admin_id = @admin_id;

END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE()
    RAISERROR(@ErrorMessage, 16, 1)
    RETURN
END CATCH
END
GO
```

```
CREATE PROCEDURE pa_client_insert (
      @name VARCHAR(50),
      @phone VARCHAR(20),
      @email VARCHAR(70),
      @username VARCHAR(30),
      @password VARCHAR(150)
) AS
BEGIN
BEGIN TRY

 --Name validation

IF @name IS NULL OR LTRIM(RTRIM(@name)) = ''
BEGIN
 RAISERROR('El nombre no corresponde a un valor válido', 16, 1)
 RETURN
END

--Phone validation
IF @phone IS NULL OR LTRIM(RTRIM(@phone)) = ''
BEGIN
 RAISERROR('El telefono no puede estar vacio', 16, 1)
 RETURN
END
ELSE IF @phone LIKE '%[^0-9]%' OR LEN(@phone) <> 8
BEGIN
 RAISERROR('El formato del teléfono no es válido', 16, 1)
 RETURN
END
ELSE IF EXISTS (SELECT 1 FROM [client_phones] WHERE phone = @phone)
BEGIN
 RAISERROR('El teléfono ya esta asociado a un cliente', 16, 1)
 RETURN
END

--Email validation

IF @email NOT LIKE '%_@_%._%'
BEGIN
 RAISERROR('El formato email no es válido', 16, 1)
 RETURN
END
ELSE IF EXISTS (SELECT 1 FROM [user] WHERE email = @email)
BEGIN
 RAISERROR('El correo ya esta asosiado a un usuario', 16, 1)
 RETURN
END

--Username validation

IF @username IS NULL OR LTRIM(RTRIM(@username)) = ''
BEGIN
 RAISERROR('El nombre de usuario no corresponde a un valor válido', 16,
1)
```

```
 RETURN;
END
ELSE IF exists (SELECT 1 FROM [user] WHERE username = @username)
BEGIN
 RAISERROR('El nombre de usuario ya se encuentra en uso', 16, 1)
 RETURN
END

--Password Validation

IF @password IS NULL OR LTRIM(RTRIM(@password)) = ''
BEGIN
RAISERROR('la contraseña no es valida', 16, 1)
 RETURN
END
ELSE IF (@password NOT LIKE '%[0-9]%' AND @password NOT LIKE
'%[!@#$^&*()]%')
BEGIN
 RAISERROR('La contraseña debe incluir un numero y un caracter especial',
16, 1)
 RETURN
END

--

INSERT INTO client([name]) VALUES (@name)
DECLARE @client_id INT = SCOPE_IDENTITY()

INSERT INTO client_phones(client_id, [phone])
values (@client_id, @phone)

INSERT INTO [user] (email, username, [password], client_id, admin_id)
values (@email, @username, @password, @client_id, null)

END TRY
BEGIN CATCH
      RAISERROR('Hubo un error en el proceso, por favor, revise los datos
ingresados', 16, 1)
      RETURN
END CATCH
END
GO


CREATE PROCEDURE pa_client_delete(
 @user_id INT
)AS
BEGIN
BEGIN TRY

 --validation
 IF NOT EXISTS(SELECT 1 FROM [user] WHERE id = @user_id AND [role] =
'CLIENT')
 BEGIN
```

```
   RAISERROR('El cliente no existe', 16, 1)
   RETURN
  END
  --
  BEGIN TRANSACTION
  DECLARE @client_id INT
  SELECT @client_id = c.id FROM client AS c INNER JOIN [user] AS u ON C.id
= u.client_id
  WHERE u.id = @user_id
   DELETE FROM client_phones WHERE client_id = @client_id;

   DELETE FROM [user] WHERE id = @user_id;

   DELETE FROM client WHERE id = @client_id;

  COMMIT TRANSACTION

END TRY
BEGIN CATCH
  DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE()
  RAISERROR(@ErrorMessage, 16, 1)
  RETURN
END CATCH
END
GO

CREATE PROCEDURE pa_client_update (
    @user_id INT,
    @new_name VARCHAR(50) = NULL,
    @new_phone VARCHAR(20) = NULL,
    @new_email VARCHAR(70) = NULL,
    @new_username VARCHAR(30) = NULL,
    @new_password VARCHAR(150) = NULL
) AS
BEGIN
BEGIN TRY
    DECLARE @client_id INT
    SELECT @client_id = c.id FROM client AS c INNER JOIN [user] AS u ON
C.id = u.client_id
    WHERE u.id = @user_id

    IF NOT EXISTS (SELECT 1 FROM [user] AS u INNER JOIN client AS c ON
u.client_id = c.id
        WHERE u.id = @user_id)
    BEGIN
        RAISERROR('El cliente no existe', 16, 1)
        RETURN
    END

    IF LTRIM(RTRIM(@new_name)) = ''
    BEGIN
        RAISERROR('El nombre no puede ser un valor vacio', 16, 1)
        RETURN
    END
    IF @new_phone IS NOT NULL
    BEGIN
        IF @new_phone LIKE '%[^0-9]%' OR LEN(@new_phone) <> 8
        BEGIN
            RAISERROR('El formato del teléfono no es válido', 16, 1)
            RETURN
        END
        ELSE IF EXISTS (SELECT 1 FROM client_phones WHERE phone =
@new_phone AND client_id <> @client_id)
        BEGIN
            RAISERROR('El teléfono ya está asociado a otro cliente', 16,
1)
            RETURN
        END
    END


    IF @new_email IS NOT NULL
    BEGIN
        IF @new_email NOT LIKE '%_@_%._%'
        BEGIN
            RAISERROR('El formato email no es válido', 16, 1)
            RETURN
        END
        ELSE IF EXISTS (SELECT 1 FROM [user] WHERE email = @new_email AND
client_id <> @client_id)
        BEGIN
            RAISERROR('El correo ya está asociado a otro usuario', 16, 1)
            RETURN
        END
    END


    IF @new_username IS NOT NULL
    BEGIN
        IF LTRIM(RTRIM(@new_username)) = ''
        BEGIN
            RAISERROR('El nombre de usuario  no puede ser un valor
vacio', 16, 1)
            RETURN
        END
        ELSE IF EXISTS (SELECT 1 FROM [user] WHERE username =
@new_username AND client_id <> @client_id)
        BEGIN
            RAISERROR('El nombre de usuario ya se encuentra en uso', 16,
1)
            RETURN
        END
    END


    IF @new_password IS NOT NULL
    BEGIN
```

```
        IF LTRIM(RTRIM(@new_password)) = ''
        BEGIN
            RAISERROR('La contraseña  no puede ser un valor vacio', 16,
1)
            RETURN
        END
        ELSE IF (@new_password NOT LIKE '%[0-9]%' OR @new_password NOT
LIKE '%[!@#$^&*()]%')
        BEGIN
            RAISERROR('La contraseña debe incluir un número y un carácter
especial', 16, 1)
            RETURN
        END
    END

    --

    UPDATE client SET name = ISNULL(@new_name, name) WHERE id =
@client_id;

    --

    IF @new_phone IS NOT NULL
    BEGIN
        IF EXISTS (SELECT 1 FROM client_phones WHERE client_id =
@client_id)
            UPDATE client_phones SET phone = @new_phone WHERE client_id =
@client_id;
        ELSE
            INSERT INTO client_phones (client_id, phone) VALUES
(@client_id, @new_phone);
    END

    UPDATE [user]SET
    email = ISNULL(@new_email, email),
    username = ISNULL(@new_username, username),
    [password] = ISNULL(@new_password, [password])
    WHERE client_id = @client_id;

END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE()
    RAISERROR(@ErrorMessage, 16, 1)
    RETURN
END CATCH
END
GO

CREATE PROCEDURE pa_extra_delete(
 @extra_id INT
)AS
BEGIN
BEGIN TRY
```

```
IF NOT EXISTS (SELECT 1 FROM extra WHERE id = @extra_id)
BEGIN
 RAISERROR ('El extra no existe', 16, 1)
 RETURN
END

 DELETE FROM extra_detail WHERE extra_id = @extra_id;

 DELETE FROM extra WHERE id = @extra_id;

END TRY
BEGIN CATCH
 RAISERROR ('El extra no ha podido eliminarse...', 16, 1)
END CATCH
END

create procedure pa_extra_details_reservation_insert(
      @person_count int,
      @extra_id int,
      @reservation_id int
) as
begin
begin try
      set nocount on
      declare @selected_reservation int
      select @selected_reservation = id from reservations where id =
@reservation_id
      if @selected_reservation is null
      begin
            raiserror('La reservación no existe', 16, 1)
            return
      end
      declare @selected_extra int
      select @selected_extra = id from extra where id = @extra_id
      if @selected_extra is null
      begin
            raiserror('El extra no existe', 16, 1)
            return
      end
      declare @limit_participants int
      select @limit_participants = isnull(count(p.id),0) + 1 from
reservations as r inner join
      tour_detail as td on r.id = td.reservation_id inner join passengers
      as p on td.id = p.tour_detail_id where td.reservation_id =
@selected_reservation
      if @person_count > @limit_participants
      begin
            raiserror('El número de participantes del extra no puede
superar el número de pasajeros', 16, 1)
            return
      end
      insert into extra_detail (person_count, extra_id, reservation_id)
      values (@person_count, @selected_extra, @selected_reservation)
end try
```

```
begin catch
    declare @ErrorMessage nvarchar(4000) = error_message();
    raiserror(@ErrorMessage,16,1);
end catch
end

CREATE PROCEDURE pa_tour_update (
    @tour_id INT,
    @new_type VARCHAR(50) = NULL,
    @new_description VARCHAR(150) = NULL,
    @new_price DECIMAL(10,2) = NULL
) AS
BEGIN
BEGIN TRY
    IF NOT EXISTS (SELECT 1 FROM tour WHERE id = @tour_id)
    BEGIN
        RAISERROR('El tour no existe', 16, 1)
        RETURN
    END

    -- Validaciones
    IF @new_type IS NOT NULL AND @new_type = ' '
    BEGIN
        RAISERROR('El tipo de tour no es válido', 16, 1)
        RETURN
    END
    IF @new_price IS NOT NULL AND @new_price <= 0
    BEGIN
        RAISERROR('El precio debe ser mayor a 0', 16, 1)
        RETURN
    END

    -- Actualización
    UPDATE tour
    SET
        type = ISNULL(@new_type, type),
        description = ISNULL(@new_description, description),
        price = ISNULL(@new_price, price)
    WHERE id = @tour_id;
END TRY
BEGIN CATCH
    RAISERROR('Error al actualizar tour', 16, 1)
END CATCH
END
GO

CREATE PROCEDURE pa_tour_search_by_id
    @tour_id INT
AS
BEGIN
BEGIN TRY

IF @tour_id IS NULL OR @tour_id <= 0
BEGIN
```

```
 RAISERROR('El id proporcionado no es válido', 16, 1);
 RETURN;
END

IF NOT EXISTS (SELECT 1 FROM tour WHERE id = @tour_id)
BEGIN
 RAISERROR ('El tour no existe', 16, 1)
 RETURN
END

SELECT
id,
[type],
[description],
price
FROM tour
WHERE @tour_id = @tour_id;

END TRY
BEGIN CATCH
 RAISERROR ('Error al buscar el tour',16, 1)
END CATCH
END
GO

REATE PROCEDURE pa_tour_search_all
AS
BEGIN
BEGIN TRY
SELECT
id,
[type],
[description],
price
FROM tour
END TRY
BEGIN CATCH
 RAISERROR('Error al obtener los tours', 16, 1);
END CATCH
END
GO

CREATE PROCEDURE pa_tour_details_reservation_update (
      @tour_detail_id INT,
      @origin VARCHAR(40),
      @destination VARCHAR(40),
      @tour_id INT,
      @reservation_id INT
) AS
BEGIN
      SET NOCOUNT ON
      BEGIN TRY
```

```
			IF NOT EXISTS (SELECT 1 FROM tour_detail WHERE id = 
@tour_detail_id)
			BEGIN
				RAISERROR('El detalle de tour no existe', 16, 1)
				RETURN
			END

			IF @origin IS NULL OR LTRIM(RTRIM(@origin)) = ''
			BEGIN
				RAISERROR('El lugar de origen no puede estar vacío', 16, 
1)
				RETURN
			END


			IF @destination IS NULL OR LTRIM(RTRIM(@destination)) = ''
			BEGIN
				RAISERROR('El lugar de destino no puede estar vacío', 
16, 1)
				RETURN
			END


			IF NOT EXISTS (SELECT 1 FROM tour WHERE id = @tour_id)
			BEGIN
				RAISERROR('El tour no existe', 16, 1)
				RETURN
			END


			IF NOT EXISTS (SELECT 1 FROM reservations WHERE id = 
@reservation_id)
			BEGIN
				RAISERROR('La reservación no existe', 16, 1)
				RETURN
			END

			UPDATE tour_detail
			SET origin = @origin,
				destination = @destination,
				tour_id = @tour_id,
				reservation_id = @reservation_id
			WHERE id = @tour_detail_id
		END TRY
		BEGIN CATCH
			DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
			RAISERROR(@ErrorMessage,16,1);
		END CATCH
END
GO

create procedure pa_tour_details_reservation_insert (
		@origin varchar(40),
		@destination varchar(40),
      @tour_id int,
      @reservation_id int
) as
begin
      set nocount on
      begin try
            if @origin is null or LTRIM(RTRIM(@origin)) = ''
            begin
                  raiserror('El lugar de origen no puede estar vacío', 16,
1)
                  return
            end
            if @destination is null or LTRIM(RTRIM(@destination)) = ''
            begin
                  raiserror('El lugar de destino no puede estar vacío',
16, 1)
                  return
            end
            declare @selected_tour int
            select @selected_tour = id from tour where id = @tour_id
            if @selected_tour is null
            begin
                  raiserror('El tour no existe', 16, 1)
                  return
            end
            declare @selected_reservation int
            select @selected_reservation = id from reservations where id
= @reservation_id
            if @selected_reservation is null
            begin
                  raiserror('La reservación no existe', 16, 1)
                  return
            end
            declare @default_supplier int = 1
            insert into tour_detail (origin, destination, tour_id,
reservation_id, supplier_id)
            values (@origin, @destination, @selected_tour,
@selected_reservation, @default_supplier)
      end try
      begin catch
            declare @ErrorMessage nvarchar(4000) = error_message();
            raiserror(@ErrorMessage,16,1);
      end catch
end
go

CREATE PROCEDURE pa_tour_details_reservation_delete (
      @tour_detail_id INT
) AS
BEGIN
      SET NOCOUNT ON
      BEGIN TRY
```

```
            IF NOT EXISTS (SELECT 1 FROM tour_detail WHERE id =
@tour_detail_id)
            BEGIN
                  RAISERROR('El detalle de tour no existe', 16, 1)
                  RETURN
            END

            DELETE FROM tour_detail WHERE id = @tour_detail_id
      END TRY
      BEGIN CATCH
            DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
            RAISERROR(@ErrorMessage,16,1);
      END CATCH
END
GO

CREATE PROCEDURE pa_tour_delete(
 @tour_id INT
)AS
BEGIN
BEGIN TRY

IF NOT EXISTS (SELECT 1 FROM tour WHERE id = @tour_id)
BEGIN
 RAISERROR ('El tour no existe', 16, 1)
 RETURN
END

 DELETE FROM tour_detail WHERE @tour_id = @tour_id;

 DELETE FROM tour WHERE id = @tour_id;

END TRY
BEGIN CATCH
 RAISERROR ('El tour no ha podido eliminarse...', 16, 1)
END CATCH
END

CREATE PROCEDURE pa_tour_insert (
    @type VARCHAR(50),
    @description VARCHAR(150),
    @price DECIMAL(10,2)
) AS
BEGIN
BEGIN TRY
    IF @type IS NULL OR @type = ' '
    BEGIN
        RAISERROR('El tipo de tour no es válido', 16, 1)
        RETURN
    END
    IF @price <= 0
    BEGIN
        RAISERROR('El precio debe ser mayor a 0', 16, 1)
        RETURN
    END

    INSERT INTO tour ([type], [description], price)
    VALUES (@type, @description, @price);
END TRY
BEGIN CATCH
    RAISERROR('Error al insertar tour', 16, 1)
END CATCH
END
GO

CREATE PROCEDURE pa_supplier_update (
    @supplier_id INT,
    @new_name VARCHAR(50) = NULL,
    @new_description VARCHAR(150) = NULL,
    @new_email VARCHAR(70) = NULL,
    @new_phone VARCHAR(20) = NULL
) AS
BEGIN
BEGIN TRY
    IF NOT EXISTS (SELECT 1 FROM supplier WHERE id = @supplier_id)
    BEGIN
        RAISERROR('El proveedor no existe', 16, 1)
        RETURN
    END

    -- Validaciones
    IF @new_name IS NOT NULL AND @new_name = ' '
    BEGIN
        RAISERROR('El nombre no corresponde a un valor válido', 16, 1)
        RETURN
    END
    IF @new_email IS NOT NULL AND @new_email NOT LIKE '%_@_%._%'
    BEGIN
        RAISERROR('El formato del email no es válido', 16, 1)
        RETURN
    END
    IF @new_phone IS NOT NULL
    BEGIN
        IF @new_phone LIKE '%[^0-9]%' OR LEN(@new_phone) <> 8
        BEGIN
            RAISERROR('El teléfono no es válido (solo 8 dígitos)', 16, 1)
            RETURN
        END
        IF EXISTS (SELECT 1 FROM supplier_phones WHERE supplier_id =
@supplier_id)
            UPDATE supplier_phones SET phone = @new_phone WHERE
supplier_id = @supplier_id;
        ELSE
            INSERT INTO supplier_phones (supplier_id, phone) VALUES
(@supplier_id, @new_phone);
    END

    -- Actualización de supplier
```

```sql
    UPDATE supplier
    SET
        name = ISNULL(@new_name, name),
        description = ISNULL(@new_description, description),
        email = ISNULL(@new_email, email)
    WHERE id = @supplier_id;
END TRY
BEGIN CATCH
    RAISERROR('Error al actualizar proveedor', 16, 1)
END CATCH
END
GO


CREATE PROCEDURE pa_supplier_delete (
    @supplier_id INT
) AS
BEGIN
BEGIN TRY
    IF NOT EXISTS (SELECT 1 FROM supplier WHERE id = @supplier_id)
    BEGIN
        RAISERROR('El proveedor no existe', 16, 1)
        RETURN
    END

    DELETE FROM supplier_phones WHERE supplier_id = @supplier_id;
    DELETE FROM tour_detail WHERE supplier_id = @supplier_id;
    DELETE FROM supplier WHERE id = @supplier_id;
END TRY
BEGIN CATCH
    RAISERROR('Error al eliminar proveedor', 16, 1)
END CATCH
END
GO

CREATE PROCEDURE pa_supplier_insert (
    @name VARCHAR(50),
    @description VARCHAR(150),
    @email VARCHAR(70),
    @phone VARCHAR(20)
) AS
BEGIN
BEGIN TRY
    IF @name IS NULL OR @name = ' '
    BEGIN
        RAISERROR('El nombre no corresponde a un valor válido', 16, 1)
        RETURN
    END
    IF @email NOT LIKE '%_@_%._%'
    BEGIN
        RAISERROR('El formato de correo no es válido', 16, 1)
        RETURN
    END
    IF @phone IS NULL OR @phone LIKE '%[^0-9]%' OR LEN(@phone) <> 8
    BEGIN
        RAISERROR('El teléfono no es válido (solo 8 dígitos)', 16, 1)
        RETURN
    END

    INSERT INTO supplier ([name], [description], [email])
    VALUES (@name, @description, @email);

    DECLARE @supplier_id INT = SCOPE_IDENTITY();

    INSERT INTO supplier_phones (supplier_id, phone)
    VALUES (@supplier_id, @phone);
END TRY
BEGIN CATCH
    RAISERROR('Error al insertar proveedor', 16, 1)
END CATCH
END
GO

CREATE PROCEDURE pa_reservation_update (
      @reservation_id INT,
      @date DATE = NULL,
      @time TIME = NULL,
      @description VARCHAR(100) = NULL,
      @user_id INT = NULL
) AS
BEGIN
SET NOCOUNT ON
BEGIN TRY

      IF NOT EXISTS (SELECT 1 FROM reservations WHERE id =
@reservation_id)
      BEGIN
            RAISERROR('La reservación no existe', 16, 1)
            RETURN
      END

      IF @date IS NOT NULL AND @time IS NOT NULL
      BEGIN
            DECLARE @reservation_datetime DATETIME2 = CAST(@date AS
DATETIME) + CAST(@time AS DATETIME)
            IF @reservation_datetime < SYSDATETIME()
            BEGIN
                  RAISERROR('La fecha y la hora que se ingresaron no son
válidas', 16, 1)
                  RETURN
            END
      END


      UPDATE reservations
      SET [date] = ISNULL(@date, [date]),
            [time] = ISNULL(@time, [time]),
            [description] = ISNULL(@description, [description]),
            [user_id] = ISNULL(@user_id, [user_id])
      WHERE id = @reservation_id

END TRY
BEGIN CATCH
      RAISERROR('Ha ocurrido un error al actualizar la reservación', 16,
1)
      RETURN
END CATCH
END
GO


CREATE PROCEDURE pa_reservation_delete (
      @reservation_id INT
) AS
BEGIN
SET NOCOUNT ON
BEGIN TRY

      IF NOT EXISTS (SELECT 1 FROM reservations WHERE id =
@reservation_id)
      BEGIN
            RAISERROR('La reservación no existe', 16, 1)
            RETURN
      END

      DELETE FROM reservations WHERE id = @reservation_id

END TRY
BEGIN CATCH
      RAISERROR('Ha ocurrido un error al eliminar la reservación', 16, 1)
      RETURN
END CATCH
END
GO


CREATE PROCEDURE pa_reservation_insert (
      @date DATE = NULL,
      @time TIME = NULL,
      @description VARCHAR(100) = NULL,
      @user_id INT = NULL
) AS BEGIN
SET NOCOUNT ON
BEGIN TRY
DECLARE @reservation_datetime  DATETIME2 = CAST(@date AS DATETIME) +
CAST(@time AS DATETIME)
      IF (@reservation_datetime) < SYSDATETIME()
      BEGIN
            RAISERROR('La fecha y la hora que se ingresaron no son
válidas', 16, 1)
            RETURN
      END
      INSERT INTO reservations ([date], [time], [description], [user_id])
      VALUES (@date, @time, @description, @user_id)
      SELECT SCOPE_IDENTITY() AS reservation_id
END TRY
BEGIN CATCH
      RAISERROR('Ha ocurrido un error en el proceso, revise los datos
nuevamente', 16, 1)
      RETURN
END CATCH
END
GO

CREATE PROCEDURE pa_passenger_update(
 @passenger_id INT,
 @name VARCHAR(50),
 @age INT
)AS
BEGIN
BEGIN TRY

 IF NOT EXISTS (SELECT 1 FROM passengers WHERE id = @passenger_id)
 BEGIN
    RAISERROR ('El pasajero no existe',16,1)
    RETURN
 END


 IF @name IS NULL OR @name = ''
 BEGIN
  RAISERROR ('El nombre no puede ser un valor vacío',16, 1)
  RETURN
 END

 IF @age <= 0
 BEGIN
   RAISERROR ('La edad debe ser un número válido',16, 1)
   RETURN
 END

 UPDATE passengers
 SET [name] = @name,
     age = @age
 WHERE id = @passenger_id;

END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
    RAISERROR(@ErrorMessage,16,1);
END CATCH
END
GO

CREATE PROCEDURE pa_passenger_delete(
 @passenger_id INT
)AS
BEGIN
BEGIN TRY

 IF NOT EXISTS (SELECT 1 FROM passengers WHERE id = @passenger_id)
 BEGIN
    RAISERROR ('El pasajero no existe',16,1)
    RETURN
 END

 DELETE FROM passengers WHERE id = @passenger_id;

END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
    RAISERROR(@ErrorMessage,16,1);
END CATCH
END
GO

--version demo
USE coco_tours_db
GO

CREATE PROCEDURE pa_passenger_insert(
 @name VARCHAR(50),
 @age INT,
 @tour_detail_id INT
)AS
BEGIN
BEGIN TRY

 IF @name IS NULL OR @name = ''
 BEGIN
  RAISERROR ('El nombre no puede ser un valor vacio',16, 1)
  RETURN
 END

 IF @age <= 0
 BEGIN
   RAISERROR ('La edad debe ser un numero valido',16, 1)
   RETURN
 END

 IF NOT EXISTS (SELECT 1 FROM tour_detail WHERE id = @tour_detail_id)
 BEGIN
    RAISERROR ('El detalle de tour no existe',16, 1)
   RETURN
 END

 INSERT INTO passengers ([name], age, tour_detail_id) VALUES (@name,
@age, @tour_detail_id);
END TRY
BEGIN CATCH
```

```
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
    RAISERROR(@ErrorMessage,16,1);
END CATCH
END
GO

CREATE PROCEDURE pa_extra_update(
 @extra_id INT,
 @new_name VARCHAR(50) = NULL,
 @new_description VARCHAR(150) = NULL,
 @new_price DECIMAL(10,2) = NULL
) AS
BEGIN
BEGIN TRY

 IF NOT EXISTS (SELECT 1 FROM extra WHERE id =@extra_id)
 BEGIN
  RAISERROR('El extra no existe...', 16, 1)
  RETURN
 END

 IF @new_name IS NOT NULL AND @new_name = ''
 BEGIN
  RAISERROR('El  nombre del extra no es valido...', 16, 1)
  RETURN
 END

 IF @new_description IS NOT NULL AND @new_description = ''
 BEGIN
   RAISERROR('La descripcion del extra no es valido...', 16, 1)
   RETURN
 END

 IF @new_price IS NOT NULL AND @new_description <= 0
 BEGIN
   RAISERROR('El precio debe ser mayor a 0', 16, 1)
   RETURN
 END

 UPDATE extra SET
 name = ISNULL(@new_name, name),
 description = ISNULL(@new_description, description),
 unit_price = ISNULL(@new_price, unit_price)
WHERE id = @extra_id

END TRY
BEGIN CATCH
    RAISERROR('No se ha logrado actualizar el extra...', 16, 1)
    RETURN
END CATCH
END
GO

CREATE PROCEDURE pa_extra_search_by_id
```

```
    @extra_id INT
AS
BEGIN
BEGIN TRY

IF @extra_id IS NULL OR @extra_id <= 0
BEGIN
 RAISERROR('El id proporcionado no es válido', 16, 1);
 RETURN;
END

IF NOT EXISTS (SELECT 1 FROM extra WHERE id = @extra_id)
BEGIN
 RAISERROR ('El extra no existe', 16, 1)
 RETURN
END

SELECT
id,
[name],
[description],
unit_price
FROM extra
WHERE @extra_id = @extra_id;

END TRY
BEGIN CATCH
 RAISERROR ('Error al buscar el extra',16, 1)
END CATCH
END
GO



CREATE PROCEDURE pa_extra_search_all
AS
BEGIN
BEGIN TRY
SELECT
id,
[name],
[description],
unit_price
FROM extra;
END TRY
BEGIN CATCH
 RAISERROR('Error al obtener los extras', 16, 1);
END CATCH
END
GO

create procedure pa_extra_details_reservation_update(
    @extra_detail_id int,
    @person_count int,
    @extra_id int
) as
begin
begin try
    set nocount on;

    declare @selected_extra_detail int, @reservation_id int;
    select @selected_extra_detail = id, @reservation_id = reservation_id
    from extra_detail where id = @extra_detail_id;

    if @selected_extra_detail is null
    begin
        raiserror('El detalle extra no existe', 16, 1);
        return;
    end


    declare @selected_extra int;
    select @selected_extra = id from extra where id = @extra_id;

    if @selected_extra is null
    begin
        raiserror('El extra no existe', 16, 1);
        return;
    end


    declare @limit_participants int;
    select @limit_participants = isnull(count(p.id),0)
    from reservations as r
    inner join tour_detail as td on r.id = td.reservation_id
    inner join passengers as p on td.id = p.tour_detail_id
    where td.reservation_id = @reservation_id;

    if @person_count > @limit_participants
    begin
        raiserror('El número de participantes del extra no puede superar
el número de pasajeros', 16, 1);
        return;
    end

    update extra_detail
    set person_count = @person_count,
        extra_id = @selected_extra
    where id = @selected_extra_detail;

end try
begin catch
    declare @ErrorMessage nvarchar(4000) = error_message();
    raiserror(@ErrorMessage,16,1);
end catch
end
GO
```

```
create procedure pa_extra_details_reservation_delete(
    @extra_detail_id int
) as
begin
begin try
    set nocount on;

    declare @selected_extra_detail int;
    select @selected_extra_detail = id from extra_detail where id =
@extra_detail_id;
g
    if @selected_extra_detail is null
    begin
        raiserror('El detalle extra no existe', 16, 1);
        return;
    end

    delete from extra_detail where id = @selected_extra_detail;
end try
begin catch
    declare @ErrorMessage nvarchar(4000) = error_message();
    raiserror(@ErrorMessage,16,1);
end catch
end
GO

--
CREATE PROCEDURE pa_backup_db
AS
BEGIN
  DECLARE @file_name NVARCHAR(200);
  SET @file_name = 'C:\Backup\coco_tours_' +
                     CONVERT(VARCHAR(20), GETDATE(), 112) + '.bak';

  BACKUP DATABASE coco_tours_db
  TO DISK = @file_name
  WITH INIT;

  PRINT 'Respaldo completado: ' + @file_name;
END;
GO

--Vistas

CREATE VIEW v_month_income AS
SELECT
  YEAR(R.date) AS anio,
  MONTH(R.date) AS mes,
  SUM(R.total) AS ingresos_totales,
  COUNT(R.id) AS cantidad_reservas
FROM reservations as R
GROUP BY YEAR(R.date), MONTH(R.date)
GO
```

```
 CREATE VIEW v_reservation_upcoming AS
 SELECT
   R.id AS reserva_id,
   C.[name] AS cliente,
   R.[date] AS fecha,
   R.total AS total
 FROM reservations AS R
 JOIN [user] AS U ON U.id = R.user_id
 JOIN client AS C ON C.id = U.client_id
 WHERE R.date >= CAST(GETDATE() AS DATE);
 GO

CREATE VIEW v_tours_popularity AS
SELECT
  T.[type],
  COUNT(TD.id) AS veces_reservado,
  SUM(R.total) AS ingresos_generados
FROM tour_detail AS TD
JOIN tour AS T ON T.id = TD.tour_id
JOIN reservations AS R ON R.id = TD.reservation_id
GROUP BY T.[type];
GO
```