Universidad Peruana Los Andes
Facultad de Ingeniería
Escuela Profesional de Ingeniería de Sistemas y Computación

Examen Final

Administración de Base de Datos

Apellidos y Nombres: ___RAMIREZ YARASCA, EDWIN_____ Código: __940817E____
Ciclo: ____V_____ Salón: _____A1_____

Enunciado 01:

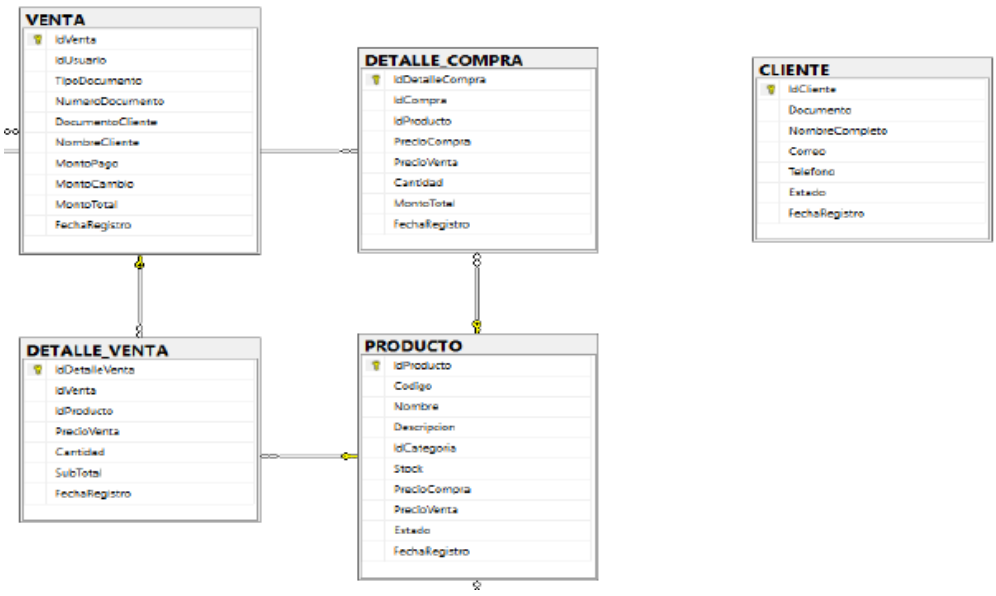
De acuerdo con la **base de datos** desarrollada en **Microsoft SQL Server**, responda las siguientes preguntas:

1) Explique que problema soluciona su base de datos

La base de datos "sistema de ventas" tiene como objetivo principal resolver problemas críticos asociados con la gestión de información relacionada con productos, clientes, ventas y el control de inventarios. En primer lugar, la centralización de los datos permite eliminar redundancias y evitar la duplicidad de registros, lo que asegura una mayor integridad y consistencia en la información. Además, el sistema automatiza procesos que suelen estar sujetos a errores humanos, como el registro de ventas, la actualización de inventarios y el cálculo de precios o descuentos, lo que mejora la precisión y eficiencia operativa. También facilita el acceso a información en tiempo real, brindando a los responsables del negocio la capacidad de tomar decisiones informadas rápidamente. Por último, al proporcionar reportes detallados, la base de datos apoya estrategias de negocio al ofrecer insights valiosos sobre el rendimiento de ventas y comportamiento de los clientes.

2) Implemente un Script para crear una **vista** para crear utilizando tres tablas

Usaremos nuestras tablas (Cliente, Venta, Producto)



SQLQuery1.sql - DE...UFSFOB7\JESU (52))*

```sql
USE DBVENTASDEMO;
GO

CREATE VIEW VistaVentasDetalle
AS
SELECT
    v.IdVenta,
    v.TipoDocumento,
    v.NumeroDocumento,
    v.NombreCliente,
    v.DocumentoCliente,
    p.Nombre AS NombreProducto,
    dv.Cantidad,
    dv.PrecioVenta,
    dv.SubTotal,
    v.FechaRegistro
FROM VENTA v
JOIN DETALLE_VENTA dv ON v.IdVenta = dv.IdVenta
JOIN PRODUCTO p ON dv.IdProducto = p.IdProducto;
GO
```

130 %

Messages

```
Commands completed successfully.

Completion time: 2024-12-21T15:25:37.3596981-05:00
```

SQLQuery2.sql - DE...UFSFOB7\JESU (65))* | SQLQuery1.sql - DE...UFSFOB7\JESU (52))*

```sql
USE DBVENTASDEMO;
GO

SELECT * FROM VistaVentasDetalle;
```

130 %

Results | Messages

	IdVenta	TipoDocumento	NumeroDocumento	NombreCliente	DocumentoCliente	NombreProducto	Cantidad	PrecioVenta	SubTotal	FechaRegistro
1	1	Boleta	00001	eerrgvr	123	Mouse Logitech G230	1	130.00	130.00	2024-12-18 20:56:06.633
2	1	Boleta	00001	eerrgvr	123	Jugo de Naranja	1	3.60	3.60	2024-12-18 20:56:06.633
3	2	Boleta	00002	abel	1122	Harina Favorita	2	3.00	6.00	2024-12-18 20:57:35.287
4	2	Boleta	00002	abel	1122	Leche Laive	6	3.40	20.40	2024-12-18 20:57:35.287

3) Implemente un Script para crear un **procedimiento almacenado** para modificar el ingreso de datos en forma secuencial

SQLQuery2.sql - DE...UFSFOB7\JESU (65))* | Ejercicio2.sql - DE...-UFSFOB7\JESU (52))

```
USE DBVENTASDEMO;
GO

CREATE PROCEDURE ModificarProductoSecuencial
    @IdProducto INT,
    @Nombre NVARCHAR(50),
    @Descripcion NVARCHAR(50),
    @Stock INT,
    @PrecioCompra DECIMAL(10,2),
    @PrecioVenta DECIMAL(10,2)
AS
BEGIN
    SET NOCOUNT ON;

    -- Validar si el producto existe
    IF EXISTS (SELECT 1 FROM PRODUCTO WHERE IdProducto = @IdProducto)
    BEGIN
        -- Actualizar el producto en forma secuencial
        BEGIN TRANSACTION;

        BEGIN TRY
            -- Actualizar nombre y descripción
            UPDATE PRODUCTO
            SET Nombre = @Nombre,
                Descripcion = @Descripcion
            WHERE IdProducto = @IdProducto;
```

SQLQuery2.sql - DE...UFSFOB7\JESU (65))* | Ejercicio2.sql - DE...-UFSFOB7\JESU (52))

```
            -- Actualizar stock
            UPDATE PRODUCTO
            SET Stock = @Stock
            WHERE IdProducto = @IdProducto;

            -- Actualizar precios
            UPDATE PRODUCTO
            SET PrecioCompra = @PrecioCompra,
                PrecioVenta = @PrecioVenta
            WHERE IdProducto = @IdProducto;

            -- Confirmar la transacción
            COMMIT TRANSACTION;
        END TRY
        BEGIN CATCH
            -- Revertir la transacción en caso de error
            ROLLBACK TRANSACTION;
            THROW;
        END CATCH
    END
    ELSE
    BEGIN
        PRINT 'El producto no existe.';
    END
END
GO
;
```

130 %

Messages

```
Commands completed successfully.

Completion time: 2024-12-21T15:36:54.0213721-05:00
```

4) Implemente un Script para crear un **disparador** para verificar el control de datos (Ejemplo: que la nota ingresada este entre 0 y 20)

SQLQuery4.sql - DE...UFSFOB7\JESU (64)) | Ejercicio4.sql - DE...-UFSFOB7\JESU (61)) | Ejercicio3.sql - D

```
USE DBVENTASDEMO;
GO

CREATE TRIGGER VerificarEstadoProducto
ON PRODUCTO
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Verificar que el campo Estado solo sea 1 o 0
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE Estado NOT IN (0, 1)
    )
    BEGIN
        -- Revertir la operación y lanzar un error
        ROLLBACK TRANSACTION;
        THROW 50000, 'El campo Estado debe ser 0 o 1.', 1;
    END
END;
GO
```

130 %

Messages

```
Commands completed successfully.

Completion time: 2024-12-21T15:42:55.2091671-05:00
```

5) Utilizando Script Crear 03 usuarios con nombres de sus compañeros y uno suyo

```
SQLQuery4.sql - DE...UFSFOB7\JESU (64))*  Ejercicio4.sql - DE...-UFSFOB7\JESU (61))  Ejercicio3.sql - DE...-UFSFOB7\JESU (65))

USE DBVENTASDEMO;
GO

CREATE LOGIN LUIS_AQUINO WITH PASSWORD = 'ContraseñaSegura123';
GO

CREATE USER LUIS_AQUINO FOR LOGIN LUIS_AQUINO;
GO

EXEC sp_addrolemember 'db_datareader', 'LUIS_AQUINO';
GO

CREATE LOGIN DAYANA_RIOS WITH PASSWORD = 'ContraseñaSegura123';
GO

CREATE USER DAYANA_RIOS FOR LOGIN DAYANA_RIOS;
GO

EXEC sp_addrolemember 'db_datareader', 'DAYANA_RIOS';
GO

CREATE LOGIN EDWIN_RAMIREZ_YARASCA WITH PASSWORD = 'ContraseñaSegura123';
GO

CREATE USER EDWIN_RAMIREZ_YARASCA FOR LOGIN EDWIN_RAMIREZ_YARASCA;
GO

EXEC sp_addrolemember 'db_datareader', 'EDWIN_RAMIREZ_YARASCA';
GO

130 %
Messages
Commands completed successfully.

Completion time: 2024-12-21T15:55:38.6464124-05:00
```

6) Utilizando un script, copiar la base de datos (creada anteriormente) y compartir en cada uno de los usuarios

```
Ejercicio6.sql - DE...-UFSFOB7\JESU (66))  Ejercicio5.sql - DE...-UFSFOB7\JESU (64))  Ejercicio4.sql - DE...-UFSFOB7\JESU (61))  Ejercicio3.sql - DE...-UFS

BACKUP DATABASE DBVENTASDEMO
TO DISK = 'C:\backups\DBVENTASDEMO.bak';
GO

RESTORE DATABASE DBVENTASDEMO_COPIA
FROM DISK = 'C:\backups\DBVENTASDEMO.bak'
WITH
    MOVE 'DBVENTASDEMO' TO 'C:\SQLData\DBVENTASDEMO_COPIA.mdf',
    MOVE 'DBVENTASDEMO_log' TO 'C:\SQLData\DBVENTASDEMO_COPIA_log.ldf';
GO


USE DBVENTASDEMO_COPIA;
GO
```

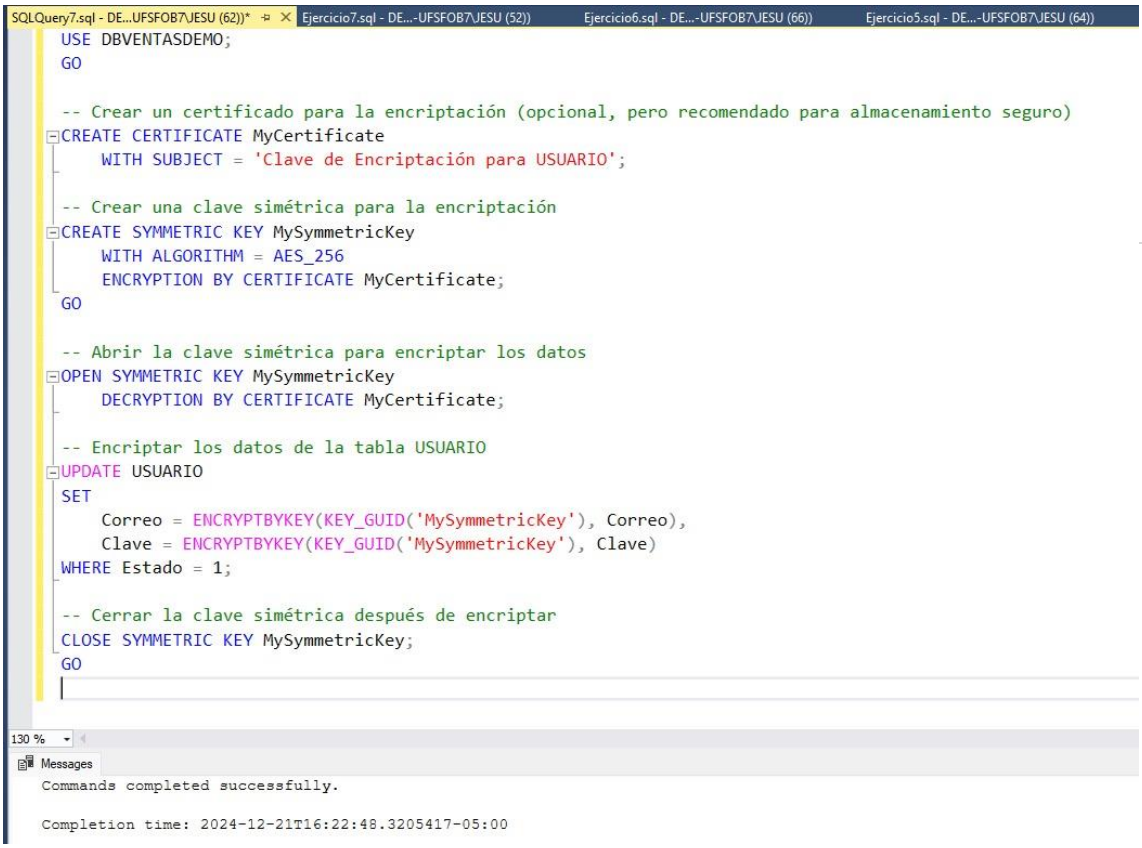
```
-- Crear usuarios en la base de datos copiada y asignarles permisos
-- Usuario LUIS AQUINO
CREATE USER LUIS_AQUINO FOR LOGIN LUIS_AQUINO;
EXEC sp_addrolemember 'db_datareader', 'LUIS_AQUINO';
EXEC sp_addrolemember 'db_datawriter', 'LUIS_AQUINO'; -- Si quieres permisos de escritura
GO

-- Usuario DAYANA RIOS
CREATE USER DAYANA_RIOS FOR LOGIN DAYANA_RIOS;
EXEC sp_addrolemember 'db_datareader', 'DAYANA_RIOS';
EXEC sp_addrolemember 'db_datawriter', 'DAYANA_RIOS'; -- Si quieres permisos de escritura
GO

-- Usuario EDWIN RAMIREZ YARASCA
CREATE USER EDWIN_RAMIREZ_YARASCA FOR LOGIN EDWIN_RAMIREZ_YARASCA;
EXEC sp_addrolemember 'db_datareader', 'EDWIN_RAMIREZ_YARASCA';
EXEC sp_addrolemember 'db_datawriter', 'EDWIN_RAMIREZ_YARASCA'; -- Si quieres permisos de escritura
GO
```

7) Utilizando un script, generar una copia de seguridad de la base de datos y compartir a cada uno de los usuarios

Ejercicio7.sql - DE...-UFSFOB7\JESU (52)) | Ejercicio6.sql - DE...-UFSFOB7\JESU (66)) | Ejercicio5.sql - DE...-UFSFOB7\JESU (64))

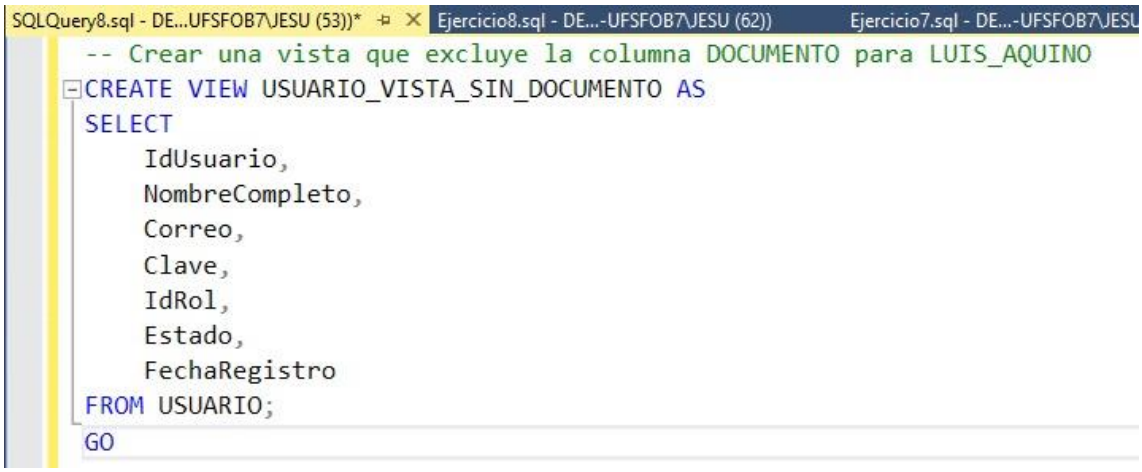
```
BACKUP DATABASE DBVENTASDEMO
TO DISK = 'C:\backups\DBVENTASDEMO.bak'
WITH FORMAT,
    MEDIANAME = 'DBVENTASDEMO_Backup',
    NAME = 'Full Backup of DBVENTASDEMO';
GO

CREATE PROCEDURE RestaurarBaseDeDatos
    @RutaCopiaBackUp NVARCHAR(255)
AS
BEGIN
    RESTORE DATABASE DBVENTASDEMO_COPIA
    FROM DISK = @RutaCopiaBackUp
    WITH
        MOVE 'DBVENTASDEMO' TO 'C:\SQLData\DBVENTASDEMO_COPIA.mdf',
        MOVE 'DBVENTASDEMO_log' TO 'C:\SQLData\DBVENTASDEMO_COPIA_log.ldf';
END;
GO

EXEC RestaurarBaseDeDatos
    @RutaCopiaBackUp = 'C:\backups\DBVENTASDEMO.bak';
GO

USE DBVENTASDEMO_COPIA;
GO

GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo TO [LUIS_AQUINO];
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo TO [DAYANA_RIOS];
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo TO [EDWIN_RAMIREZ_YARASCA];
GO
```

8) Utilizando un script, encriptar una de las tablas para que no se puedan ver los datos

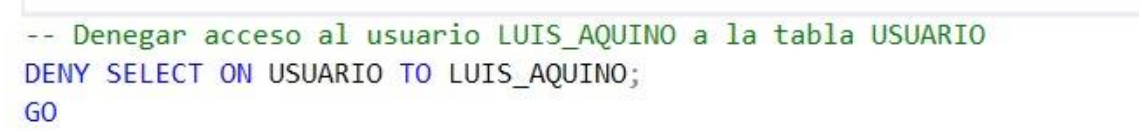


9) Utilizando un script, aplique la seguridad a nivel de columna, restringiendo el acceso a la columna DNI de la tabla empleado en el usuario con nombre de su compañero

Primero, creamos una vista que excluye la columna DOCUMENTO para el usuario LUIS_AQUINO. Este usuario no podrá ver esa columna al consultar la vista.



Para restringir el acceso a la columna DOCUMENTO, denegamos el acceso directo a la tabla USUARIO y solo otorgamos acceso a la vista.



Permitir que los administradores o usuarios con privilegios accedan a la columna DOCUMENTO

```sql
-- Otorgar permisos al usuario LUIS_AQUINO sobre la vista sin la columna DOCUMENTO
GRANT SELECT ON USUARIO_VISTA_SIN_DOCUMENTO TO LUIS_AQUINO;
GO

-- Otorgar permisos a otros usuarios para acceder a la columna DOCUMENTO
GRANT SELECT ON USUARIO TO EDWIN_RAMIREZ_YARASCA;
GO
```


10) Utilizando un script, implementé seguridad a nivel de columna restringiendo el acceso a una de las columnas de una tabla.

SQLQuery9.sql - DE...UFSFOB7\JESU (54))* | Ejercicio9.sql - DE...-UFSFOB7\JESU (53)) | Ejercicio8.sql - DE...-UFSFOB7\JESU (62))

```sql
-- Crear una vista que excluye la columna DOCUMENTO
CREATE VIEW USUARIO_VISTA_RESTRINGIDA AS
SELECT
    IdUsuario,
    NombreCompleto,
    Correo,
    Clave,
    IdRol,
    Estado,
    FechaRegistro
FROM USUARIO;
GO

-- Denegar acceso a la columna DOCUMENTO para el usuario LUIS_AQUINO
DENY SELECT ON USUARIO(DOCUMENTO) TO LUIS_AQUINO;
GO

-- Otorgar permisos de SELECT sobre la vista sin la columna DOCUMENTO
GRANT SELECT ON USUARIO_VISTA_RESTRINGIDA TO LUIS_AQUINO;
GO
```

11) Utilizando un script, realice el cifrado transparente de datos (TDE) para una las tablas.

Crear una clave maestra de base de datos

Crear un certificado para el cifrado

Habilitar el cifrado de la base de datos

Verificar el estado del cifrado

Realizar una copia de seguridad del certificado y la clave

```
SQLQuery10.sql - D...UFSFOB7\JESU (70))*   Ejercicio10.sql - D...-UFSFOB7\JESU (54))   Ejercicio9.sql - DE...-UF

-- Crear una clave maestra de base de datos (si aún no existe)
CREATE DATABASE ENCRYPTION KEY;
GO

-- Crear un certificado para el cifrado de la base de datos
CREATE CERTIFICATE TDE_Certificate
WITH SUBJECT = 'Certificado para TDE';
GO

-- Habilitar TDE en la base de datos DBVENTASDEMO
ALTER DATABASE DBVENTASDEMO
SET ENCRYPTION ON;
GO

-- Verificar el estado de TDE en la base de datos
SELECT name, is_encrypted
FROM sys.databases
WHERE name = 'DBVENTASDEMO';
GO

-- Realizar una copia de seguridad del certificado
BACKUP CERTIFICATE TDE_Certificate
TO FILE = 'C:\Backup\TDE_Certificate.cer'
WITH PRIVATE KEY (
    FILE = 'C:\Backup\TDE_Certificate_PrivateKey.pvk',
    ENCRYPTION BY PASSWORD = 'MiContraseñaSegura'
);
GO
```

12) Utilizando un script, configure el usuario con el nombre de su compañero para otorgar permisos de SELECT, INSERT, UPDATE y DELETE en la base de datos.

```
SQLQuery12.sql - D...UFSFOB7\JESU (51))*   Ejercicio11.sql - D...-UFSFOB7\JESU (70))   Ejercicio10.sql - D...-UFSFOB7\JESU (54))   Ejerci
USE DBVENTASDEMO;
GO


CREATE USER ANGELINE FOR LOGIN ANGELINE;
GO

GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo TO ANGELINE;
GO

130 %
Messages

Completion time: 2024-12-21T16:52:18.7580012-05:00
```

13) Utilizando un script, configure la auditoría para el seguimiento y registro de acciones en la base de datos

Ejercicio13.sql - D...-UFSFOB7\JESU (67)) | Ejercicio12.sql - D...-UFSFOB7\JESU (51)) | Ejercicio11.sql - D...-UFSFOB7\JESU (70)) | Ejercicio10.sql -

```
-- Crear la auditoría para la base de datos
CREATE SERVER AUDIT MiAuditoria
    TO FILE (FILEPATH = 'C:\AuditoríaLogs\')
    WITH (QUEUE_DELAY = 1000, ON_FAILURE = CONTINUE);
GO

-- Activar la auditoría
ALTER SERVER AUDIT MiAuditoria
    WITH (STATE = ON);
GO

-- Crear una especificación de auditoría para auditar acciones en la base de datos
CREATE SERVER AUDIT SPECIFICATION MiEspecificacionDeAuditoria
    FOR SERVER AUDIT MiAuditoria
    ADD (DATABASE_OBJECT_PERMISSION_CHANGE_GROUP),
    ADD (DATABASE_PRINCIPAL_CHANGE_GROUP),
    ADD (LOGIN_CHANGE_PASSWORD_GROUP),
    ADD (FAILED_LOGIN_GROUP)
    WITH (STATE = ON);
GO

-- Consultar los eventos de auditoría desde el archivo
SELECT *
FROM fn_get_audit_file('C:\AuditoríaLogs\*.sqlaudit', NULL, NULL);
GO
```

130 %

Results | Messages

```
(0 rows affected)

Completion time: 2024-12-21T16:56:41.0289295-05:00
```

14) Utilizando un script, configure de la memoria y el disco duro

Habilitar las opciones avanzadas

Primero, necesitas ejecutar el siguiente comando para habilitar las opciones avanzadas en SQL Server:

SQLQuery14.sql - D...UFSFOB7\JESU (55))* | Ejercicio13.sql - D...-UFSFOB7\JESU (67)) | Ejercici

```
-- Habilitar opciones avanzadas
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
GO

EXEC sp_configure 'max server memory (MB)', 8192;
RECONFIGURE;
GO

EXEC sp_configure 'min server memory (MB)', 2048;
RECONFIGURE;
GO

EXEC sp_configure 'max server memory (MB)';
EXEC sp_configure 'min server memory (MB)';
GO
```

130 %

Results | Messages

	name	minimum	maximum	config_value	run_value
1	max server memory (MB)	128	2147483647	8192	8192

	name	minimum	maximum	config_value	run_value
1	min server memory (MB)	0	2147483647	2048	2048

15) Utilizando un script, genere una copia de seguridad de la base de datos

SQLQuery15.sql - D...UFSFOB7\JESU (58))* | Ejercicio14.sql - D...-UFSFOB7\JESU (55)) | Ejer

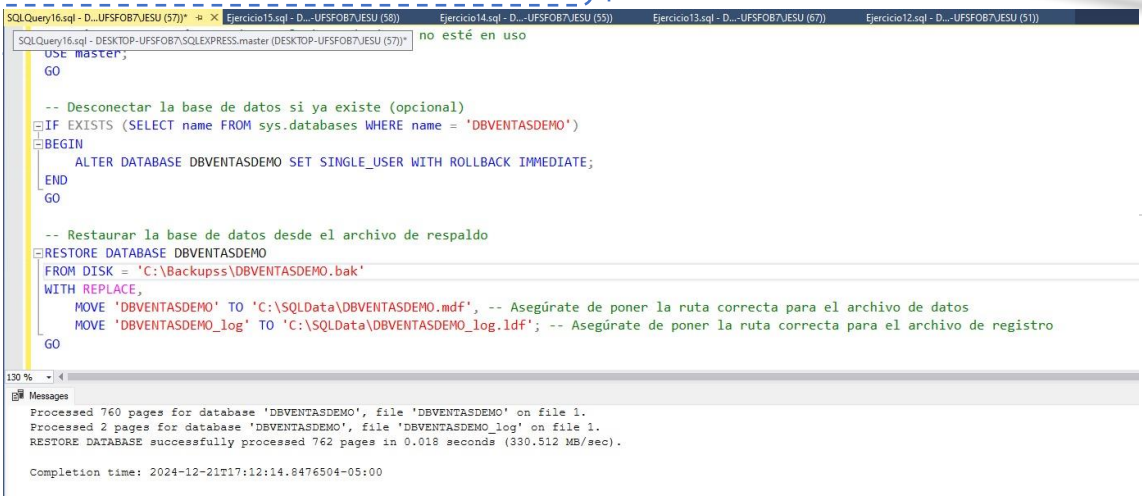
```
BACKUP DATABASE DBVENTASDEMO
TO DISK = 'C:\Backups\DBVENTASDEMO.bak'
WITH FORMAT,
    MEDIANAME = 'DBVENTASDEMOBackup',
    NAME = 'Full Backups of DBVENTASDEMO';
GO

-- Comprobar si el archivo de respaldo existe
EXEC xp_fileexist 'C:\Backups\DBVENTASDEMO.bak';
```

130 %

Results | Messages

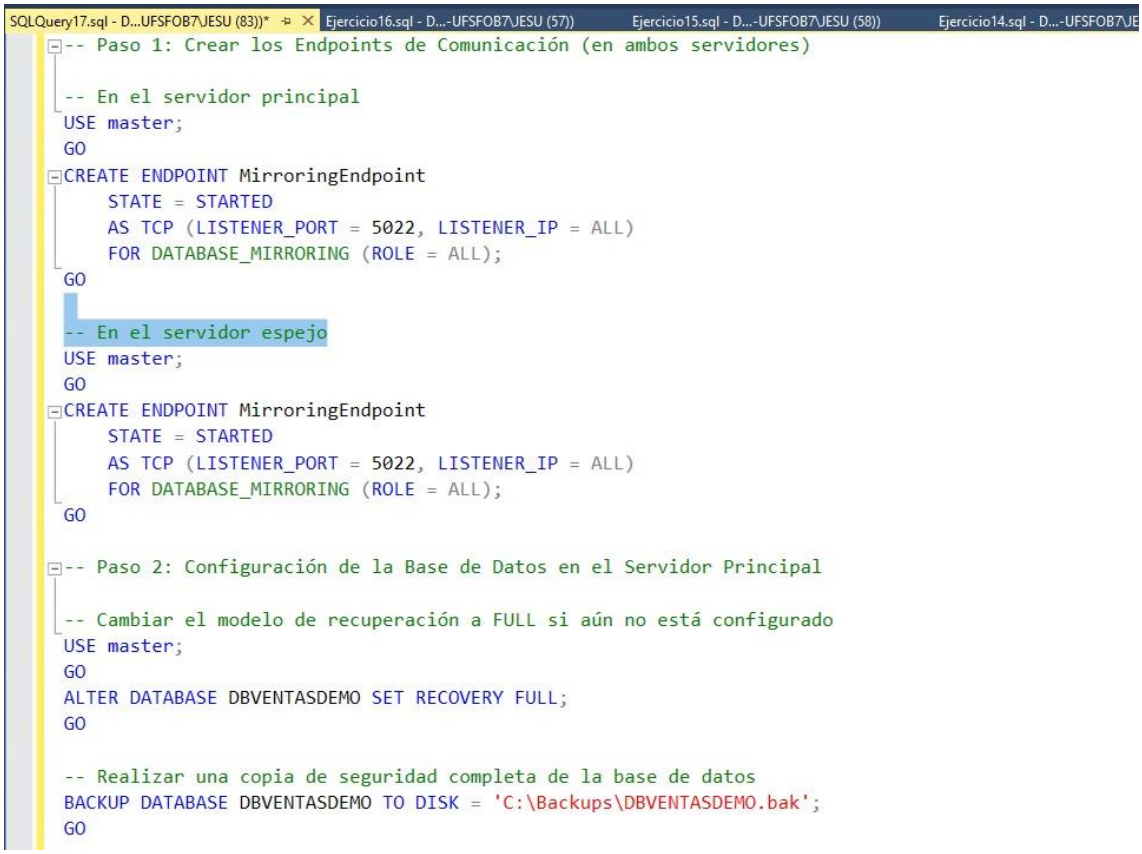
	File Exists	File is a Directory	Parent Directory Exists
1	1	0	1

16) Utilizando un script, genere la restauración de la base de datos



```
SQLQuery16.sql - D...UFSFOB7\JESU (57))*   Ejercicio15.sql - D...-UFSFOB7\JESU (58))   Ejercicio14.sql - D...-UFSFOB7\JESU (55))   Ejercicio13.sql - D...-UFSFOB7\JESU (67))   Ejercicio12.sql - D...-UFSFOB7\JESU (51))
SQLQuery16.sql - DESKTOP-UFSFOB7\SQLEXPRESS.master (DESKTOP-UFSFOB7\JESU (57))*
USE master;
GO

-- Desconectar la base de datos si ya existe (opcional)
IF EXISTS (SELECT name FROM sys.databases WHERE name = 'DBVENTASDEMO')
BEGIN
    ALTER DATABASE DBVENTASDEMO SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
END
GO

-- Restaurar la base de datos desde el archivo de respaldo
RESTORE DATABASE DBVENTASDEMO
FROM DISK = 'C:\Backupss\DBVENTASDEMO.bak'
WITH REPLACE,
    MOVE 'DBVENTASDEMO' TO 'C:\SQLData\DBVENTASDEMO.mdf', -- Asegúrate de poner la ruta correcta para el archivo de datos
    MOVE 'DBVENTASDEMO_log' TO 'C:\SQLData\DBVENTASDEMO_log.ldf'; -- Asegúrate de poner la ruta correcta para el archivo de registro
GO

130 %
Messages
Processed 760 pages for database 'DBVENTASDEMO', file 'DBVENTASDEMO' on file 1.
Processed 2 pages for database 'DBVENTASDEMO', file 'DBVENTASDEMO_log' on file 1.
RESTORE DATABASE successfully processed 762 pages in 0.018 seconds (330.512 MB/sec).

Completion time: 2024-12-21T17:12:14.8476504-05:00
```

17) Utilizando un script, cree un espejo de la base de datos



```
SQLQuery17.sql - D...UFSFOB7\JESU (83))*   Ejercicio16.sql - D...-UFSFOB7\JESU (57))   Ejercicio15.sql - D...-UFSFOB7\JESU (58))   Ejercicio14.sql - D...-UFSFOB7\JE
-- Paso 1: Crear los Endpoints de Comunicación (en ambos servidores)

-- En el servidor principal
USE master;
GO
CREATE ENDPOINT MirroringEndpoint
    STATE = STARTED
    AS TCP (LISTENER_PORT = 5022, LISTENER_IP = ALL)
    FOR DATABASE_MIRRORING (ROLE = ALL);
GO

-- En el servidor espejo
USE master;
GO
CREATE ENDPOINT MirroringEndpoint
    STATE = STARTED
    AS TCP (LISTENER_PORT = 5022, LISTENER_IP = ALL)
    FOR DATABASE_MIRRORING (ROLE = ALL);
GO

-- Paso 2: Configuración de la Base de Datos en el Servidor Principal

-- Cambiar el modelo de recuperación a FULL si aún no está configurado
USE master;
GO
ALTER DATABASE DBVENTASDEMO SET RECOVERY FULL;
GO

-- Realizar una copia de seguridad completa de la base de datos
BACKUP DATABASE DBVENTASDEMO TO DISK = 'C:\Backups\DBVENTASDEMO.bak';
GO
```

```
SQLQuery17.sql - D...UFSFOB7\JESU (83))*  Ejercicio16.sql - D...-UFSFOB7\JESU (57))  Ejercicio15.sql - D...-UFSFOB7\JESU (58))  Ejercicio14.sql - D...
GO

-- Configurar la base de datos en el servidor principal para el espejo
ALTER DATABASE DBVENTASDEMO SET PARTNER = 'TCP://<IP_SERVIDOR_ESPEJO>:5022';
GO

-- Paso 3: Restaurar la Base de Datos en el Servidor Espejo

-- Restaurar la copia de seguridad en el servidor espejo en modo NORECOVERY
USE master;
GO
RESTORE DATABASE DBVENTASDEMO FROM DISK = 'C:\Backups\DBVENTASDEMO.bak' WITH NORECOVERY;
GO

-- Configurar la base de datos en el servidor espejo para que sea el espejo
ALTER DATABASE DBVENTASDEMO SET PARTNER = 'TCP://<IP_SERVIDOR_PRINCIPAL>:5022';
GO

-- Paso 4: Configurar el Modo de Alta Seguridad (opcional)

-- Configurar el modo de alta seguridad para el espejo
USE master;
GO
ALTER DATABASE DBVENTASDEMO SET SAFETY FULL;
GO

-- Paso 5: Verificar el estado del espejo
SELECT database_id, name, state_desc, recovery_model_desc
FROM sys.databases
WHERE name = 'DBVENTASDEMO';
GO
```

18) Utilizando un script, realice la replicación de bases de datos

```
SQLQuery18.sql - D...UFSFOB7\JESU (59))*  Ejercicio17.sql - D...-UFSFOB7\JESU (83))  Ejercicio16.sql - D...-UFSFOB7\JESU (57))  Ejercicio15.sql - D...-UF
-- Paso 1: Activar la replicación en el servidor publicador
USE master;
GO
-- Habilitar la base de datos como publicador
EXEC sp_replicationdboption
    @dbname = 'DBVENTASDEMO',
    @optname = 'publish',
    @value = 'true';
GO

-- Paso 2: Configurar la base de datos de distribución
EXEC sp_adddistributor
    @distributor = 'ServidorPublicador',  -- Nombre del servidor publicador
    @password = 'ContraseñaDistribuidor'; -- Contraseña del distribuidor
GO

-- Paso 3: Crear la base de datos de distribución
EXEC sp_adddistributiondb
    @database = 'distribution',
    @security_mode = 1;  -- Modo de seguridad para la base de datos de distribución
GO

-- Paso 4: Crear el publicador
EXEC sp_addpublisher
    @publisher = 'ServidorPublicador',  -- Nombre del servidor publicador
    @distribution_db = 'distribution'; -- Base de datos de distribución
GO

-- Paso 5: Configurar el servidor suscriptor
USE master;
GO
EXEC sp_addsubscriber
    @subscriber = 'ServidorSuscriptor';  -- Nombre del servidor suscriptor
GO
```

SQLQuery18.sql - D...UFSFOB7\JESU (59))* | Ejercicio17.sql - D...-UFSFOB7\JESU (83)) | Ejercicio16.sql - D...-UFSFOB7\JESU (57)) | Ejercicio15.sql - D...-UFSFO

```
-- Paso 6: Crear la publicación en el servidor publicador
USE DBVENTASDEMO;
GO
EXEC sp_addpublication
    @publication = 'DBVENTASDEMO_Publication',  -- Nombre de la publicación
    @publication_type = 'transactional',  -- Tipo de replicación: transaccional
    @description = 'Replicación transaccional de la base de datos DBVENTASDEMO',
    @sync_method = 'native',
    @article = 'all';  -- Publicar todas las tablas
GO

-- Paso 7: Crear artículos de la publicación (tablas a replicar)
EXEC sp_addarticle
    @publication = 'DBVENTASDEMO_Publication',
    @article = 'PRODUCTO',
    @source_object = 'PRODUCTO',
    @type = 'logbased',
    @description = 'Tabla PRODUCTO';
GO

EXEC sp_addarticle
    @publication = 'DBVENTASDEMO_Publication',
    @article = 'CLIENTE',
    @source_object = 'CLIENTE',
    @type = 'logbased',
    @description = 'Tabla CLIENTE';
GO

-- Paso 8: Crear la suscripción en el servidor suscriptor
EXEC sp_addsubscription
    @publication = 'DBVENTASDEMO_Publication',
    @subscriber = 'ServidorSuscriptor',
    @subscription_type = 'push',  -- Tipo de suscripción: push (el publicador envía los datos)
    @sync_type = 'automatic';  -- Sincronización automática
GO
```

SQLQuery18.sql - D...UFSFOB7\JESU (59))* | Ejercicio17.sql - D...-UFSFOB7\JESU (83)) | Ejercicio16.sql - D...-UFSFOB7\JESU (57)) | Ejercicio15.sq

```
-- Paso 9: Configurar el Agente de Replicación para distribución
USE distribution;
GO
EXEC sp_adddistpublisher
    @publisher = 'ServidorPublicador',
    @distribution_db = 'distribution';
GO

-- Paso 10: Configurar el agente de replicación
EXEC sp_adddistributiondb
    @database = 'distribution';
GO

-- Paso 11: Iniciar la replicación
EXEC sp_startpublication_snapshot
    @publication = 'DBVENTASDEMO_Publication';
GO
```

19) Explique que es Always On Availability Groups

Always On Availability Groups es una característica avanzada de SQL Server que se utiliza para proporcionar una solución de alta disponibilidad y recuperación ante desastres. Permite agrupar un conjunto de bases de datos en un grupo de disponibilidad, ofreciendo réplicas primarias y secundarias. La réplica primaria maneja las operaciones de lectura y escritura, mientras que las réplicas secundarias se utilizan para respaldo, operaciones de solo lectura, y failover automático o manual en caso de que la réplica primaria falle.

Las réplicas secundarias pueden configurarse como sincrónicas o asincrónicas, asegurando integridad de datos en tiempo real o con un retraso aceptable, dependiendo de las necesidades del negocio. Además, las aplicaciones pueden aprovechar estas réplicas para reducir la carga de la réplica primaria mediante la ejecución de consultas de solo lectura en los nodos secundarios.

20) Explique que es Log Shipping

Log Shipping es una técnica de recuperación ante desastres que permite transferir periódicamente las copias de los registros de transacciones desde un servidor principal hacia uno o más servidores secundarios. Este método funciona mediante tres pasos básicos: realizar una copia de seguridad de los registros en el servidor primario, copiarlos a los servidores secundarios, y restaurarlos en dichos servidores.

Este enfoque asegura que las bases de datos en los servidores secundarios se mantengan sincronizadas con las del servidor principal. Aunque Log Shipping no ofrece failover automático como Always On Availability Groups, es una solución simple y confiable que permite mantener réplicas consistentes de la base de datos en escenarios de recuperación ante desastres.