

“Año de la recuperación y consolidación de la economía peruana”

UNIVERSIDAD NACIONAL DE CAJAMARCA

FACULTAD DE INGENIERÍA

E. P. DE INGENIERÍA DE SISTEMAS



Propuesta de BD para la tienda “Comercial Rita”

Curso

Base de Datos II

Docente

Ing. Jeiner Stanly Huamán Cruzado

Alumnos

Herrera Arias, Sarah Daniela Fernanda

Limay Rodríguez, Adriana Anthonela

Ocas Ruiz, Arnold Michell

Pérez Briceño, Darick André

Valdiviezo Zavaleta, Jesús Arturo

Ciclo

IV

27 de enero de 2025

ÍNDICE

ÍNDICE.....	1
I. INTRODUCCIÓN.....	3
II. RESUMEN	4
III. OBJETIVOS.....	5
A. OBJETIVO GENERAL	5
B. OBJETIVOS ESPECÍFICOS	5
1) <i>Objetivo específico 1:</i>	5
2) <i>Objetivo específico 2:</i>	5
3) <i>Objetivo específico 3:</i>	5
4) <i>Objetivo específico 4:</i>	5
5) <i>Objetivo específico 5:</i>	5
IV. MARCO TEÓRICO.....	6
A. TRANSACT-SQL (T-SQL).....	6
1) <i>Características de Transact-SQL (T-SQL)</i>	6
2) <i>Ventajas de Transact-SQL (T-SQL)</i>	6
3) <i>Funciones de Transact-SQL (T-SQL)</i>	7
B. ANÁLISIS Y DISEÑO DE BASES DE DATOS	7
1) <i>Análisis de la Base de Datos</i>	8
2) <i>Diseño de la base de datos</i>	8
C. SEGURIDAD Y OPTIMIZACIÓN EN BASES DE DATOS	9
1) <i>Concepto de Seguridad en Base de Datos</i>	9
2) <i>Roles de usuarios</i>	10
3) <i>Procedimientos Almacenados</i>	10
4) <i>Vistas</i>	11
5) <i>Funciones</i>	11
6) <i>Triggers</i>	11
D. POLÍTICAS DE BACKUPS.....	12
1) <i>Backups</i>	12
2) <i>Política de Respaldo</i>	12
3) <i>Beneficios de una Política de Backup</i>	15
E. ALMACENAMIENTO EN NUBE	15
1) <i>Ventajas</i>	15
2) <i>Funcionamiento</i>	16
3) <i>Tipos</i>	16

4) <i>Requisitos</i>	16
5) <i>Casos de uso</i>	17
V. DESARROLLO DEL PROYECTO	18
A. DISEÑO DE BASE DE DATOS.....	18
1) <i>Análisis de Requisitos Para la Realización de la Base de Datos</i>	18
2) <i>Diseño del modelo relacional de la tienda “Comercial Rita”</i>	19
B. IMPLEMENTACIÓN EN TRANSACT – SQL	19
1) <i>Script de Tablas</i>	19
2) <i>Registros</i>	21
3) <i>Procedimientos Almacenados</i>	23
4) <i>Funciones</i>	29
5) <i>Vistas</i>	32
6) <i>Triggers</i>	37
7) <i>Roles de Usuario</i>	42
8) <i>Script completo de la Base de Datos “Comercial Rita”</i>	44
C. AUTOMATIZACIÓN DE BACKUPS.....	44
1) <i>Backup Completo</i>	44
2) <i>Backup Diferencial</i>	44
3) <i>Backup Transaccional</i>	45
4) <i>Guardado en Amazon S3</i>	45
D. REPOSITORIO EN GITHUB	46
VI. CONCLUSIONES	47
VII. RECOMENDACIONES	48
VIII. REFERENCIAS	49

I. INTRODUCCIÓN

En el presente trabajo hablaremos de la era digital actual, donde la gestión eficiente de la información se ha convertido en un factor crítico para el éxito empresarial. Las empresas de Cajamarca, en su proceso de modernización y adaptación a las nuevas tecnologías, enfrentan el desafío de transformar sus sistemas tradicionales de manejo de datos, tradicionalmente basados en hojas de cálculo, hacia soluciones más robustas y seguras que permitan una gestión integral de su información.

En este contexto, el presente proyecto se centra en el diseño e implementación de un sistema de gestión de base de datos empresarial utilizando SQL Server Community, una solución que busca revolucionar la manera en que las empresas locales manejan su información, esta iniciativa no solo representa una actualización tecnológica, sino que constituye un paso fundamental hacia la optimización de procesos y el fortalecimiento de la seguridad de datos empresariales, donde la implementación de este sistema responde a necesidades críticas identificadas en el entorno empresarial local, donde la utilización de hojas de cálculo como principal herramienta de gestión de datos ha demostrado limitaciones significativas en términos de seguridad, eficiencia y escalabilidad. El nuevo sistema, desarrollado con SQL Server Community, promete no solo superar estas limitaciones, sino también proporcionar una plataforma robusta que garantice la integridad y disponibilidad de la información a través de soluciones automatizadas de respaldo en la nube.

El proyecto aborda dos componentes fundamentales: primero, el diseño e implementación de una base de datos personalizada que se adapte a las necesidades específicas del negocio, incluyendo la creación de estructuras de datos optimizadas, mecanismos de seguridad y herramientas de gestión eficientes; y segundo, el desarrollo de una solución automatizada de respaldos que garantice la protección y disponibilidad continua de la información mediante su almacenamiento seguro en la nube utilizando servicios como Amazon S3.

Esta transformación digital representa no solo una mejora tecnológica, sino también un cambio fundamental en la manera en que las empresas locales gestionan y protegen su información, preparándolas para enfrentar los desafíos del mercado actual y futuro con herramientas más eficientes y seguras

II. RESUMEN

El presente proyecto tiene como objetivo el diseño e implementación de una base de datos en SQL Server Community para la tienda de frutos secos "Comercial Rita", permitiendo una gestión eficiente del inventario, ventas, proveedores, clientes y empleados. Para garantizar la seguridad, integridad y disponibilidad de la información, se han definido estrategias de respaldo, almacenamiento en la nube y medidas de seguridad dentro de la base de datos, para lo cual se ha considerado que, para la seguridad de los datos, se utilizará Amazon S3 como solución de almacenamiento en la nube, en donde vamos a realizar backups automatizados de la base de datos. Esta estrategia permite proteger la información contra pérdidas accidentales, fallos del sistema o ataques informáticos, asegurando que siempre exista una copia de seguridad accesible y segura. La utilización de S3 garantiza durabilidad, alta disponibilidad y redundancia, optimizando costos y evitando la dependencia exclusiva del almacenamiento local, adicionalmente, dentro de la base de datos, se implementarán triggers como una medida de seguridad para evitar alteraciones no autorizadas en la información.

El diseño de la base de datos se fundamenta en un análisis de requisitos, con el objetivo de optimizar los procesos de control de stock, seguimiento de transacciones y generación de reportes de ventas y costos., esta implementación se llevará a cabo en Transact-SQL, definiendo estructuras relacionales con restricciones de integridad y claves foráneas para garantizar la coherencia de los datos.

Además, todos estos datos han sido importados desde archivos CSV y están disponibles en nuestro repositorio de GitHub, junto con los scripts SQL completos. Cabe mencionar que este enfoque integral permite una administración confiable, segura y escalable de la información de la tienda, facilitando la toma de decisiones estratégicas.

III. OBJETIVOS

A. Objetivo General

El objetivo general de nuestra investigación es diseñar e implementar un sistema de gestión de base de datos empresarial utilizando SQL Server y Transact-SQL (T-SQL) que optimice el manejo de información, mejore la seguridad de los datos y automatice los respaldos en la nube.

B. Objetivos Específicos

- 1) **Objetivo específico 1:** Crear un modelo de base de datos adaptado a las necesidades de la empresa “Comercial Rita”, definiendo tablas, relaciones, restricciones, índices y otros elementos clave.
- 2) **Objetivo específico 2:** Utilizar procedimientos almacenados, vistas, triggers y funciones para mejorar la eficiencia y seguridad de las operaciones con datos.
- 3) **Objetivo específico 3:** Establecer estrategias de seguridad para proteger la integridad y confidencialidad de la información.
- 4) **Objetivo específico 4:** Desarrollar un sistema automatizado que realice respaldos periódicos de la base de datos y los almacene en un servicio en la nube (S3), garantizando disponibilidad y recuperación en caso de fallos.
- 5) **Objetivo específico 5:** Aplicar mejores prácticas para mejorar la velocidad de consulta y la eficiencia de la base de datos mediante el uso de índices y optimización de consultas SQL.

IV. MARCO TEÓRICO

A. Transact-SQL (T-SQL)

Transact-SQL (T-SQL) es una extensión del lenguaje SQL utilizada en Microsoft SQL Server. Permite la manipulación y gestión de bases de datos relacionales mediante la ejecución de consultas y la programación de lógica avanzada dentro del servidor [1].

1) *Características de Transact-SQL (T-SQL)*

- a. Incluye estructuras de control de flujo, variables, cursores y otros elementos que no están en SQL puro [1].
- b. Permite encapsular lógica de negocio dentro de procedimientos y funciones reutilizables.
- c. Facilita la ejecución automática de acciones en respuesta a eventos en la base de datos.
- d. Permite definir permisos y roles de usuarios a nivel granular.
- e. T-SQL incorpora estructuras de control como BEGIN, END, IF...ELSE, WHILE, GOTO, RETURN, BREAK y CONTINUE, que permiten la ejecución condicional y la iteración dentro de los scripts [1] [2].
- f. Proporciona comandos como BEGIN TRANSACTION, COMMIT y ROLLBACK para asegurar la integridad de los datos [1] [2].
- g. Permite la declaración y asignación de variables locales utilizando DECLARE, SET y SELECT, lo que facilita el almacenamiento temporal de datos durante la ejecución de scripts [1] [2].

2) *Ventajas de Transact-SQL (T-SQL)*

- a. **Eficiencia en la Ejecución:** Al ejecutar código directamente en el servidor, reduce la transferencia de datos entre el cliente y el servidor, mejorando la eficiencia de las operaciones [3].
- b. **Seguridad Mejorada:** Permite definir permisos y roles a nivel granular, controlando el acceso a los datos y protegiendo la integridad de la información [3].
- c. **Facilidad de Mantenimiento:** La encapsulación de lógica en procedimientos almacenados y funciones facilita el mantenimiento y la actualización del código [3].

- d. Optimización del Rendimiento:* La capacidad de manejar transacciones y utilizar índices mejora el rendimiento de las consultas y operaciones en la base de datos [3].

3) *Funciones de Transact-SQL (T-SQL)*

Además de las funciones integradas de SQL Server, los usuarios pueden definir funciones utilizando T-SQL.

Los tipos de funciones T-SQL incluyen:

- Funciones de agregación, que operan sobre una colección de valores, pero devuelven un valor de resumen.
- Funciones de clasificación, que devuelven un valor de clasificación para cada fila dentro de una partición.
- Funciones de conjunto de filas, que devuelven un objeto que puede usarse como referencia de tabla en instrucciones SQL.
- Funciones escalares, que operan sobre un único valor y devuelven un único valor.

SQL Server también admite funciones analíticas en T-SQL para representar tareas analíticas complejas. Estas funciones analíticas permiten a los profesionales de TI realizar análisis comunes, como clasificaciones, percentiles, promedios móviles y sumas acumulativas que se expresarán en una única declaración SQL [4][5].

Además, T-SQL permite la creación de tablas, vistas y objetos temporales, lo que facilita la organización y manipulación de datos. La escritura de consultas avanzadas en T-SQL incluye el uso de funciones de ventana, que permiten realizar cálculos sobre un conjunto de filas relacionadas con la fila actual, proporcionando una forma eficiente de realizar agregaciones y análisis de datos [5] [6].

Transact-SQL es una herramienta poderosa que extiende las capacidades de SQL estándar, ofreciendo funcionalidades avanzadas para el manejo eficiente de datos en entornos empresariales. Su integración de características procedurales y de control de flujo permite a los desarrolladores implementar lógica compleja directamente en la base de datos, optimizando tanto el rendimiento como la seguridad de las aplicaciones.

B. Análisis y Diseño de Bases de Datos

El diseño inicial de una base de datos debe abordar soluciones al problema identificado, aunque puede ajustarse para adaptarse a las necesidades de los usuarios [7]. El análisis y diseño

es crucial en el desarrollo de sistemas, aplicando metodologías y mejores prácticas para garantizar una base de datos funcional, escalable y fácil de mantener.

1) *Análisis de la Base de Datos*

- a. *Análisis de Requisitos e identificación del propósito de la Base de Datos:*** El análisis de requisitos es fundamental en el diseño de bases de datos, ya que define su estructura y funcionalidad. Incluye entrevistas con usuarios y la revisión de sistemas existentes para comprender la información necesaria, los patrones de consulta y los problemas frecuentes [7]. Este análisis permite elaborar una declaración de objetivos clara que guía el diseño y facilita la toma de decisiones alineadas con las necesidades identificadas [8].
- b. *Estructuras de las bases de datos relacionales:*** El diseño de una base de datos relacional inicia con el modelo de datos, que describe los datos, sus relaciones y restricciones. Su objetivo es ofrecer una estructura lógica independiente de la implementación física, facilitando su manejo y mantenimiento [9].

2) *Diseño de la base de datos*

- a. *Diseño Conceptual:*** El modelo de datos conceptual define las entidades y relaciones necesarias para cumplir los requisitos comerciales, mientras que el diseño lógico las traduce a tablas relacionales, utilizando claves primarias y foráneas. La normalización organiza los datos para evitar redundancia, mejorar la integridad y optimizar el rendimiento de la base de datos [9].
- b. *Diseño Lógico:*** El modelo de datos lógico es un paso intermedio entre el modelo conceptual y el modelo físico, detallando las relaciones de datos sin depender de un sistema de gestión específico. A diferencia del modelo conceptual, que se enfoca en los requisitos comerciales, el modelo lógico organiza y resuelve redundancias y relaciones complejas, sirviendo como puente hacia la implementación técnica [10].
- c. *Diseño Físico:*** El modelo de datos físico define la implementación de la base de datos, especificando tablas, columnas, tipos de datos, claves primarias y foráneas, así como las relaciones entre tablas. Se adapta al DBMS utilizado y puede incluir desnormalización según los requisitos. Los pasos básicos son convertir entidades

en tablas, relaciones en claves externas y atributos en columnas, ajustando el modelo según restricciones físicas [11].

d. Otros Diseños

- **Diseño ascendente:** El diseño ascendente comienza con el modelo físico de la base de datos, derivando los modelos lógicos y conceptuales a partir de él. Es un enfoque flexible y adaptativo que responde a las fuentes de datos existentes y las necesidades cambiantes, aunque puede ser caótico e inconsistente debido a la falta de una visión coherente de los datos y sus relaciones [12].
- **Diseño híbrido:** El diseño híbrido de bases de datos combina enfoques de arriba hacia abajo y de abajo hacia arriba, integrando modelos conceptuales y físicos. Busca alinear los requisitos de datos con las tecnologías disponibles, permitiendo planificación y flexibilidad, aunque requiere una coordinación compleja entre las etapas del proceso [12].

C. Seguridad y Optimización en Bases de Datos

1) Concepto de Seguridad en Base de Datos

La seguridad de la base de datos se refiere a la variedad de herramientas, controles y medidas diseñadas para establecer y preservar la confidencialidad, integridad y disponibilidad de la base de datos [13].

a. Prácticas de Seguridad

- **Seguridad física:** Ya sea que su servidor de base de datos esté en las instalaciones o en un centro de datos en la nube, debe estar ubicado dentro de un entorno seguro y con control del clima.
- **Controles administrativos y de acceso a la red:** El número mínimo y práctico de usuarios debería tener acceso a la base de datos, y sus permisos deberían estar restringidos a los niveles mínimos necesarios para que puedan realizar su trabajo [13].
- **Cifrado:** Todos los datos, deben protegerse con el mejor cifrado mientras están en reposo y en tránsito. Todas las claves de cifrado deben manejarse de acuerdo con las directrices de mejores prácticas [13].

- **Seguridad de copia de seguridad:** Todas las copias de seguridad, copias o imágenes de la base de datos deben estar sujetas a los mismos controles de seguridad que la propia base de datos [13].

b. Controles y políticas: Las políticas de seguridad de bases de datos deben alinearse con los objetivos comerciales, como proteger la propiedad intelectual y complementar las políticas de ciberseguridad y seguridad en la nube. Es clave asignar responsabilidades para auditar controles, establecer programas de capacitación, realizar pruebas de penetración y evaluar vulnerabilidades en apoyo a estas políticas [13].

2) *Roles de usuarios*

Los roles de usuario gestionan el acceso y las acciones en una base de datos, asignando permisos específicos para que los usuarios interactúen solo con los datos relevantes. Mientras el administrador tiene acceso total, otros usuarios necesitan roles personalizados, lo que limita su acceso según sus responsabilidades. Esto protege la integridad y confidencialidad de los datos y mejora la seguridad al reducir riesgos de accesos no autorizados [14].

3) *Procedimientos Almacenados*

Un procedimiento almacenado de SQL Server es un grupo de una o varias instrucciones Transact-SQL [15]. Como ventajas de los procedimientos, tenemos:

- Tráfico de red reducido:** Ejecutar comandos en un solo lote de código disminuye el tráfico entre el servidor y el cliente [15].
- Mayor seguridad:** Permiten a usuarios sin permisos directos realizar operaciones a través de procedimientos, protegiendo los objetos subyacentes y simplificando la gestión de permisos [15].
- Reutilización del código:** Pueden devolver un valor de estado para indicar el éxito o error de la operación, facilitando la depuración [15].
- Rendimiento mejorado:** La compilación inicial de un procedimiento crea un plan de ejecución reutilizable, lo que reduce el tiempo de procesamiento en ejecuciones posteriores [15].

4) *Vistas*

Una vista es una tabla virtual generada dinámicamente mediante una consulta, que combina y presenta datos de tablas subyacentes, incluso de diferentes bases de datos o servidores. Es útil para integrar información, personalizar la presentación de datos y simplificar consultas complejas [16].

Las vistas son útiles como mecanismo de seguridad, permitiendo acceso controlado a datos sin otorgar permisos directos sobre las tablas subyacentes. También facilitan compatibilidad con versiones anteriores, optimizan el rendimiento al gestionar datos entre servidores y particionarlos, y ofrecen flexibilidad en el manejo de información en entornos complejos [16].

5) *Funciones*

Las funciones definidas por el usuario de SQL Server son rutinas que aceptan parámetros, realizan una acción, como un cálculo complejo, y devuelven el resultado de esa acción como un valor. El valor devuelto puede ser un valor escalar único o un conjunto de resultados [17]. Entre sus ventajas, tenemos:

- **Programación modular.** Puede crear la función una vez, almacenarla en la base de datos y llamarla desde el programa tantas veces como desee [17].
- **Ejecución más rápida.** Las funciones mejoran el rendimiento al reutilizar planes en caché, evitando recompilaciones constantes, lo que acelera su ejecución. Mientras que las funciones CLR son más eficientes para cálculos, manipulación de cadenas y lógica de negocio, las de Transact-SQL son ideales para tareas intensivas de acceso a datos [17].
- **Reducción del tráfico de red.** Una operación que filtra datos basándose en restricciones complejas que no se puede expresar en una sola expresión escalar se puede expresar como una función. La función se puede invocar luego en la cláusula WHERE para reducir el número de filas que se envían al cliente [17].

6) *Triggers*

Un desencadenador es un procedimiento almacenado que se ejecuta automáticamente en respuesta a eventos específicos en el servidor de bases de datos. Permiten automatizar procesos y realizar operaciones para mantener la coherencia de los datos. Los desencadenadores DML se activan ante eventos de manipulación de datos

como INSERT, UPDATE o DELETE, independientemente de si las filas se ven afectadas. Los desencadenadores DDL responden a eventos de definición de datos como CREATE, ALTER o DROP. Los desencadenadores LOGON se ejecutan al establecerse la sesión de un usuario. Permite múltiples desencadenadores para una misma instrucción. [18].

D. Políticas de Backups

1) *Backups*

Un respaldo (backup) se refiere a una copia de los datos originales que se almacena en un lugar seguro, su importancia radica en que, en caso de una pérdida de datos, se pueda recuperar la información, minimizando el impacto en la empresa. Además, es necesario para cumplir con las regulaciones de privacidad y seguridad de datos [19].

Una política de respaldo puede no llegar a satisfacer los requisitos de todos los usuarios si no se implementa correctamente. Por ejemplo, una política que funcione bien para un sistema con un solo usuario puede ser inadecuada para un sistema que sirve a cientos de usuarios. De igual manera, una política desarrollada para un sistema donde los datos pueden modificarse frecuentemente sería ineficaz para un sistema con datos que cambian esporádicamente, por ello es fundamental, en estos casos, seguir una estrategia de copia de seguridad robusta. Sin importar cuál sea la estrategia de copia de seguridad más adecuada para un sitio, es crucial que exista una y que se realicen copias de seguridad con frecuencia y de manera regular. Lo más importante es que cada usuario puede determinar la mejor política de copia de seguridad para su sistema [20].

2) *Política de Respaldo*

- a. **Definir qué datos se deben respaldar:** Antes de establecer una política de respaldo, es esencial identificar qué datos requieren protección, esto puede abarcar información de clientes, registros financieros, detalles de proyectos y cualquier otro dato crítico para la organización [21].
- b. **Determinar la frecuencia de los backups:** La frecuencia de los respaldos variará según la cantidad de datos generados por la empresa y su importancia, normalmente, se aconseja realizar copias de seguridad diaria o semanalmente. Por ejemplo, si tienes datos que cambian cada hora y no puedes permitirte perderlos,

hacer respaldos diarios no sería adecuado, en ese caso, deberías considerar una frecuencia de una hora o menos. Con Dongee Backup, se puede configurar copias de seguridad con la periodicidad que se desee, incluso en intervalos de menos de una hora [21].

- c. *Establecer un plan de recuperación de desastres:*** Ante una eventual pérdida de datos, es crucial contar con un plan de recuperación de desastres que permita a la empresa reanudar sus operaciones rápidamente, este plan debe contemplar la estrategia de respaldo, los pasos a seguir para la recuperación de datos y las personas responsables de ejecutar cada fase del plan [21].
- d. *Implementar diferentes métodos de backup:*** Se aconseja emplear variados métodos de respaldo para asegurar la protección de los datos, esto puede abarcar respaldos en disco, respaldos en la nube y respaldos en cinta [21].
- e. *Realizar pruebas periódicas:*** Es esencial llevar a cabo pruebas periódicas para verificar que los respaldos funcionan correctamente y que los datos pueden recuperarse en caso de pérdida, estas pruebas deben realizarse, al menos, una vez al año [21].
- f. *Relacionar respaldos según el cambio en los datos***

 - ***Respaldos diarios:*** Realizar copias de seguridad lo más seguido posible, si la entidad no tiene un servidor dedicado, respalda manualmente al inicio o final del día, indicando en el respaldo la fecha. Si la entidad dispone de un servidor dedicado, lo ideal es crear una tarea programada que realice automáticamente esta tarea durante horarios no laborales como, por ejemplo, la noche. Puede ser necesario solicitar asistencia de personal informático para esta configuración [21].
 - ***Respaldos especiales:*** Si se va a realizar operaciones de modificación, borrado o alta de registros de forma masiva, es apropiado realizar un respaldo especial antes de llevar a cabo estas tareas para revertir cualquier error [21].
 - ***Protección adicional:*** Realiza un respaldo antes de someter el equipo a cualquier cambio de hardware, enviar el equipo temporalmente a otra ubicación o cuando terceros vayan a trabajar en él [21].

- g. *Mantener respaldos en múltiples ubicaciones:*** El respaldo debe almacenarse en diferentes ubicaciones para proteger los datos contra desastres locales, esto puede implicar mover copias de seguridad a medios externos como CD/DVD, pen drives, o almacenamiento en la nube. Es crucial que los respaldos sean retirados del mismo equipo o disco rígido para evitar la pérdida total en caso de fallo físico, ataque de virus o robo del equipo [21].
- h. *Designar responsables y controles:*** Definir claramente quién es responsable de realizar y verificar los respaldos es vital, pues dicha persona debe estar identificada con nombre y apellido, no solo por área o departamento. Establecer controles para asegurar que los respaldos se hagan efectivamente es fundamental para evitar omisiones [21].
- i. *Garantizar acceso y revisión de respaldos:*** Es necesario verificar periódicamente que los respaldos se están realizando correctamente y en buenas condiciones, esto incluye comprobar que los soportes físicos, como CD, DVD, o pen drives, sean operables y legibles. Asimismo, proporcionar acceso a los respaldos para los usuarios responsables de los datos es esencial para asegurar su pronta recuperación en caso de necesidad [21].
- j. *Realizar simulacros de recuperación:*** Llevar a cabo simulacros de recuperación de datos en un entorno controlado ayuda a verificar la efectividad de los respaldos y la capacidad de restaurar información cuando sea necesario, esto es una práctica recomendada para asegurar la fiabilidad del proceso de respaldo [21].
- k. *Evaluar y ajustar la política de respaldo regularmente:*** Realiza evaluaciones periódicas de las políticas de respaldo y ajustarlas según los cambios en la infraestructura, el volumen de datos o amenazas emergentes, mantener una política de respaldo robusta y adaptada a las necesidades específicas de la organización es fundamental para asegurar la protección continua de los datos [21].
- l. *Cuidado de la unidad de disco:*** Es importante cuidar el hardware que almacena los datos para evitar fallos inesperados, reemplazar los discos rígidos antes de que alcancen el fin de su vida útil puede evitar la pérdida completa de datos y situaciones inconvenientes, consultar sobre la vida útil de los discos y planificar su sustitución puede ser una barrera adicional de protección [21].

3) *Beneficios de una Política de Backup*

- a. ***Máxima Flexibilidad:*** Mantener una copia principal de los datos de respaldo en las instalaciones para una recuperación rápida es prudente. Además, contar con una copia secundaria en la nube para recuperación ante desastres brinda flexibilidad para enfrentar interrupciones no deseadas [21].
- b. ***Protección y Cumplimiento Normativo de los Datos:*** Proteger los datos contra la eliminación accidental, la corrupción y ataques maliciosos es esencial. Las empresas pueden aprovechar la encriptación de datos y la seguridad de la gestión de acceso disponibles en la nube, asegurando beneficios cuando se mantiene el cumplimiento normativo [21].
- c. ***Operaciones Optimizadas en Cuanto a Costos:*** Una política de respaldo que extiende la protección de datos a la nube ayuda a liberar inversiones vinculadas a infraestructura local, cambiando los gastos de capital iniciales por gastos operativos más económicos. Este enfoque, junto con el respaldo de almacenamiento de objetos en la nube, ofrece durabilidad superior y limita los costos potenciales de pérdida [21].

E. Almacenamiento en Nube

Es un método de almacenamiento externo gestionado por un tercero y presenta una alternativa al almacenamiento local de datos, esta tecnología permite guardar de manera segura archivos y documentos importantes en una base de datos remota, eliminando la necesidad de almacenarlos en el disco duro de la computadora o en otros dispositivos de almacenamiento personal [22].

1) *Ventajas*

Entre las ventajas del almacenamiento en la nube se destaca la eliminación de la necesidad de poseer físicamente un dispositivo de almacenamiento, como es el caso de las memorias USB, reduciendo así el riesgo de perder datos irremplazables, además, el almacenamiento en la nube simplifica el intercambio de contenido, ya que solo basta con compartir una carpeta, para que colegas o colaboradores puedan acceder instantáneamente a sus contenidos. Otro importante beneficio del almacenamiento en la nube es el ahorro en costos, pues resulta mucho más económico y eficiente adquirir el espacio de almacenamiento necesario en la nube mediante una tarifa nominal, en

comparación con la compra y mantenimiento de un amplio espacio de almacenamiento local [22].

Asimismo, el almacenamiento en la nube ofrece una mayor flexibilidad, permitiendo acceder a los datos desde cualquier lugar con conexión a internet, lo que facilita el trabajo remoto y la colaboración en tiempo real. También proporciona una mayor seguridad, ya que los proveedores de servicios en la nube implementan medidas avanzadas de protección de datos, como cifrado y copias de seguridad automáticas, garantizando la integridad y disponibilidad continua de la información almacenada [22].

2) *Funcionamiento*

En la computación en la nube, los servicios de almacenamiento en la nube operan como una red interconectada de servidores de datos, utilizada colectivamente para compartir y acceder a archivos a través de diferentes dispositivos, los proveedores de almacenamiento en la nube se encargan de poseer y mantener estos servidores externos que forman parte de la red en sus propios centros de datos, los usuarios tienen la posibilidad de cargar archivos a estos servidores y acceder a sus datos almacenados en la nube mediante un sitio web, una aplicación de escritorio o una aplicación móvil [22].

3) *Tipos*

Existen tres tipos principales de almacenamiento en la nube [23]:

- a. ***Almacenamiento de objetos:*** Ideal para grandes volúmenes de datos no estructurados, permite almacenar datos en su formato original y personalizar metadatos para facilitar el acceso y análisis.
- b. ***Almacenamiento de archivos:*** Organiza los datos en un formato jerárquico de carpetas y archivos, comúnmente utilizado en servidores NAS.
- c. ***Almacenamiento en bloques:*** Utilizado para aplicaciones empresariales que requieren almacenamiento dedicado y de baja latencia, similar al almacenamiento conectado directamente.

4) *Requisitos*

Al considerar el almacenamiento en la nube, es fundamental tener en cuenta [23]:

- a. ***Durabilidad y disponibilidad:*** Los datos se almacenan de forma redundante en múltiples dispositivos, mejorando la durabilidad y disponibilidad.

- b. Seguridad:** Es esencial que los datos estén cifrados y que existan controles de acceso robustos para proteger la información.

5) *Casos de uso*

El almacenamiento en la nube tiene múltiples aplicaciones, incluyendo [23]:

- a. Análisis y lagos de datos:** Permite crear lagos de datos seguros y escalables para análisis de grandes volúmenes de información.
- b. Copias de seguridad y recuperación de desastres:** Ofrece una solución de bajo costo y alta durabilidad para la protección de datos.
- c. Pruebas y desarrollo de software:** Facilita la creación y gestión de entornos de almacenamiento para pruebas y desarrollo, migración de datos a la nube, facilita la transferencia de grandes volúmenes de datos a la nube.
- d. Conformidad:** Permite cumplir con regulaciones y normativas de seguridad de datos.
- e. Almacenamiento de aplicaciones nativas en la nube:** *Soporta* aplicaciones que utilizan tecnologías como contenedorización y microservicios.
- f. Archivo:** Proporciona soluciones de archivo que ofrecen durabilidad y accesibilidad a largo plazo.
- g. Almacenamiento en la nube híbrida:** *Conecta* aplicaciones locales con almacenamiento en la nube para optimizar costos y administración.
- h. Almacenamiento de base de datos:** Ofrece alto rendimiento y escalabilidad para bases de datos transaccionales.
- i. ML e IoT:** Facilita el procesamiento y análisis de datos para aplicaciones de machine learning e Internet de las cosas.

V. DESARROLLO DEL PROYECTO

A. Diseño de Base de Datos

1) *Análisis de Requisitos Para la Realización de la Base de Datos*

El principal objetivo de la base de datos es gestionar de manera eficiente y organizada la información relacionada con el inventario, las ventas, los proveedores, los empleados, y los clientes de la tienda de frutos secos "Comercial Rita". Esto permitirá optimizar los procesos de control de stock, realizar un seguimiento detallado de las transacciones y facilitar la toma de decisiones.

a. Gestión de inventario

- Registrar los productos disponibles con sus respectivas características (nombre, descripción, precio, etc.).
- Controlar las existencias actuales y registrar movimientos de entrada y salida de productos.
- Generar alertas automáticas cuando el stock de un producto alcance el nivel crítico de reabastecimiento.

b. Gestión de proveedores

- Registrar información de los proveedores (nombre, dirección, etc.).
- Vincular productos específicos a sus respectivos proveedores.
- Realizar un seguimiento de los pedidos realizados a cada proveedor.

c. Gestión de ventas

- Registrar todas las transacciones realizadas, incluyendo la fecha, productos vendidos, cantidad, y el monto total.
- Relacionar las ventas con los clientes recurrentes para crear un historial.

d. Gestión de clientes

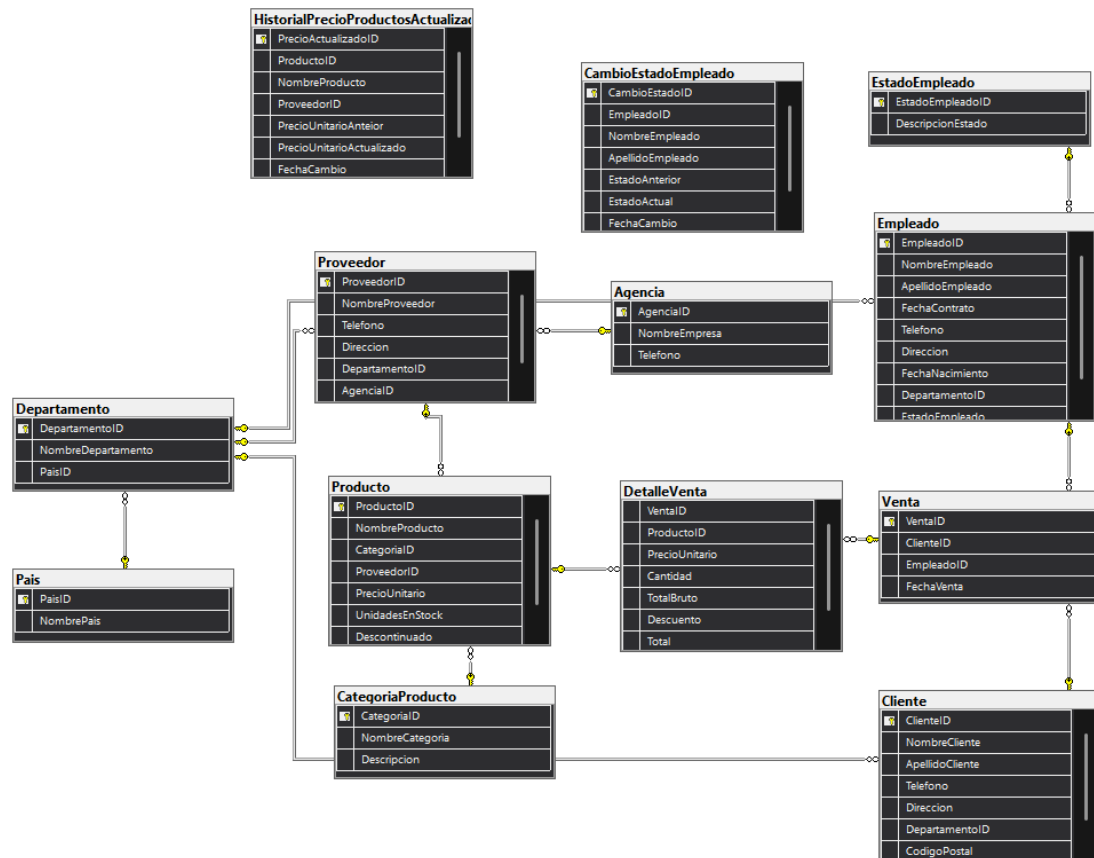
- Registrar clientes frecuentes con información básica (nombre, contacto, historial de compras, etc.).
- Ofrecer descuentos o promociones personalizadas basadas en el historial de compras.

e. Generación de reportes

- Reportes de ventas mensuales y anuales.
- Reportes de productos más vendidos.

- Reportes de costos y márgenes de ganancia.

2) Diseño del modelo relacional de la tienda “Comercial Rita”



B. Implementación en Transact – SQL

1) Script de Tablas

```
USE ComercialRita

CREATE TABLE Producto ( -- CREAMOS TABLA PRODUCTOS
    ProductoID INT PRIMARY KEY IDENTITY(1,1) NOT NULL, -- ID de producto, va
    NombreProducto NVARCHAR(50) NOT NULL,
    CategoriaID INT, FOREIGN KEY (CategoriaID) REFERENCES
    CategoriaProducto(CategoriaID),
    ProveedorID INT, FOREIGN KEY (ProveedorID) REFERENCES
    Proveedor(ProveedorID),
    PrecioUnitario MONEY DEFAULT 0, CHECK(PrecioUnitario >= 0), -- Precio mayor
    e igual que 0 por defecto.
    UnidadesEnStock DECIMAL(10,2) DEFAULT 0, CHECK(UnidadesEnStock >= 0), --
    Stock mayor e igual que 0 por defecto.
    Descontinuado BIT -- 0 (No descontinuado), 1 (Descontinuado).
)

CREATE TABLE CategoriaProducto (
    CategoriaID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    NombreCategoria NVARCHAR(50) NOT NULL,
    Descripcion NVARCHAR(255)
```

```

)

CREATE TABLE Proveedor (
    ProveedorID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    NombreProveedor NVARCHAR(50) NOT NULL,
    Telefono NVARCHAR(15),
    Direccion NVARCHAR(100),
    DepartamentoID INT, FOREIGN KEY (DepartamentoID) REFERENCES
Departamento(DepartamentoID),
    AgenciaID INT, FOREIGN KEY (AgenciaID) REFERENCES Agencia(AgenciaID)
)

CREATE TABLE Empleado (
    EmpleadoID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    NombreEmpleado NVARCHAR(50) NOT NULL,
    ApellidoEmpleado NVARCHAR(50) NOT NULL,
    FechaContrato DATETIME,
    Telefono NVARCHAR(15),
    Direccion NVARCHAR(30),
    FechaNacimiento DATETIME,
    DepartamentoID INT, FOREIGN KEY (DepartamentoID) REFERENCES
Departamento(DepartamentoID),
    EstadoEmpleado INT
)

CREATE TABLE Cliente (
    ClienteID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    NombreCliente NVARCHAR(50) NOT NULL,
    ApellidoCliente NVARCHAR(50) NOT NULL,
    Telefono NVARCHAR(15),
    Direccion NVARCHAR(30),
    DepartamentoID INT, FOREIGN KEY (DepartamentoID) REFERENCES
Departamento(DepartamentoID),
    CodigoPostal NVARCHAR(10),
)

CREATE TABLE Departamento (
    DepartamentoID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    NombreDepartamento NVARCHAR(20) NOT NULL,
    PaisID INT NOT NULL, FOREIGN KEY (PaisID) REFERENCES Pais(PaisID)
)

CREATE TABLE Pais (
    PaisID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    NombrePais NVARCHAR(20) NOT NULL
)

CREATE TABLE Venta (
    VentaID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    ClienteID INT NOT NULL, FOREIGN KEY (ClienteID) REFERENCES
Cliente(ClienteID),
    EmpleadoID INT NOT NULL, FOREIGN KEY (EmpleadoID) REFERENCES
Empleado(EmpleadoID),
    FechaVenta DATETIME NOT NULL,
)

CREATE TABLE DetalleVenta (
    VentaID INT NOT NULL, FOREIGN KEY (VentaID) REFERENCES Venta(VentaID),
    ProductoID INT NOT NULL, FOREIGN KEY (ProductoID) REFERENCES
Producto(ProductoID),
    PrecioUnitario INT NOT NULL,

```

```

        Cantidad DECIMAL (10,2) NOT NULL,
        TotalBruto DECIMAL (10,2) NOT NULL,
        Descuento DECIMAL (10,2) NOT NULL,
        Total DECIMAL (10,2) NOT NULL,
    )

CREATE TABLE Agencia (
    AgenciaID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    NombreEmpresa NVARCHAR(20) NOT NULL,
    Telefono NVARCHAR(15), CHECK(Telefono >= 9)
)

CREATE TABLE EstadoEmpleado (
    EstadoEmpleadoID INT IDENTITY (1,1) PRIMARY KEY,
    DescripcionEstado NVARCHAR(20),
)

```

2) Registros

Se han importado los datos desde archivos csv. a la base de datos “ComercialRita”. Los archivos se pueden encontrar en el repositorio “sql-server-database-proyecto-Comercial-Rita”, en GitHub.

a. Tabla “Producto”

ProductoID	NombreProducto	CategorialID	ProveedorID	PrecioUnitario	UnidadesEnStock	Descontinuado
1	Almendras naturales	1	1	44,00	5.00	0
2	Almendras tostadas	1	1	46,00	5.00	0
3	Castañas de primera	1	1	48,00	6.00	0
4	Cashews tostados	1	1	44,00	7.00	0
5	Pistachos tostados	1	1	56,00	4.00	0
6	Pecanas con cascara	1	1	34,00	5.00	0
7	Pecanas peladas	1	1	74,00	5.00	0
8	Pasas rubias grandes	2	2	32,00	8.00	0
9	Pasas rubias pequeñas	2	2	24,00	10.00	0
10	Pasas morenas importadas	2	2	17,00	9.00	0

b. Tabla “CategoriaProducto”

CategorialID	NombreCategoria	Descripcion
1	Frutos Secos	
2	Frutos Deshidratados	
3	Frutas Procesadas	
4	Especias	
5	Condimentos	
6	Harinas	
7	Legumbres	
8	Hierbas y Tisanas	
9	Semillas	
10	Cereales	

c. Tabla “Proveedor”

ProveedorID	NombreProveedor	Telefono	Direccion	DepartamentoID	AgencialID
1	FrutoLar		1	1	1
2	FlowersFood		1	1	1
3	Nutra Estevia		1	1	1
4	Liz Conde		2	1	1

d. Tabla “Agencia”

	AgencialID	NombreEmpresa	Telefono
1	1	GilGal Cargo	NULL

e. Tabla “Empleado”

EmpleadoID	Nombre...	Apellido...	FechaContrato	Telefono	Direccion	FechaNacimiento	Dep...	EstadoEmpleado
1	Pablo	Herrera	2024-03-25 0...	955967803	Jr. Juncos ...	1987-01-07 00:0...	2	1
2	Camila	Herrera	2024-03-26 0...		Jr. Tulipanes	2007-04-04 00:0...	2	1

f. Tabla EstadoEmpleado

EstadoEmpleadoID	DescripcionEstado
1	Activo
2	Descanso
3	Desempleado

g. Tabla “Cliente”

ClientelID	NombreCliente	ApellidoCliente	Telefono	Direccion	DepartamentoID	CodigoPostal
1	Anthonela	Limay	NULL		2	
2	Arnold	Ocas	NULL		2	
3	Arturo	Valdiviezo	NULL		2	
4	Darick	Pérez	NULL		2	
5	Sarah	Herrera	NULL		2	

h. Tabla “Venta”

VentalID	ClientelID	EmpleadoID	FechaVenta
1	1	1	2024-12-01 00:00:00.000
2	3	2	2024-12-02 00:00:00.000
3	4	2	2024-12-03 00:00:00.000
4	1	1	2024-12-04 00:00:00.000
5	4	1	2024-12-05 00:00:00.000
6	3	2	2024-12-06 00:00:00.000
7	2	1	2024-12-07 00:00:00.000
8	1	2	2024-12-08 00:00:00.000
9	3	2	2024-12-09 00:00:00.000
10	1	2	2024-12-10 00:00:00.000

i. Tabla “DetalleVenta”

VentalID	ProductoID	PrecioUnitario	Cantidad	TotalBruto	Descuento	Total
1	2	46	0.25	11.50	0.00	11.50
1	5	56	0.25	14.00	0.00	14.00
1	7	74	0.25	18.50	0.00	18.50
2	69	40	0.50	20.00	0.00	20.00
2	70	7	1.00	7.00	0.00	7.00
2	71	8	1.00	8.00	0.00	8.00
3	21	60	0.50	30.00	0.00	30.00

j. Tabla “Pais”

	PaisID	NombrePais
1	1	Peru

k. Tabla “Departamento”

	DepartamentoID	NombreDepartamento	PaisID
1	1	Lima	1
2	2	Cajamarca	1

3) Procedimientos Almacenados

a. Insertar Cliente

```

CREATE PROCEDURE sp_InsertarCliente
    @NombreCliente nvarchar(50),
    @ApellidoCliente nvarchar(50),
    @Direccion nvarchar(30),
    @NombreDepartamento nvarchar(20),
    @NombrePais nvarchar(20),
    @Telefono nvarchar(15),
    @CodigoPostal nvarchar(10)
as
begin
    begin try
        begin transaction;

        --recupera el id del pais
        declare @PaisID int;
        select @PaisID = PaisID
        from Pais
        where NombrePais = @NombrePais;

        --si no existe el pais lo inserta en la tabla pais
        if @PaisID is null
        begin
            insert into Pais (NombrePais)
            values (@NombrePais);

            set @PaisID = SCOPE_IDENTITY(); --devuelve la ultima
            identidad insertada en el mismo ambito
        end

        --recupera el id del departamento
        declare @DepartamentoID int;
        select @DepartamentoID = DepartamentoID
        from Departamento
        where NombreDepartamento = @NombreDepartamento
            and PaisID = @PaisID;

        if @DepartamentoID is null
        begin
            insert into Departamento (NombreDepartamento,PaisID)
            values (@NombreDepartamento,@PaisID);

            set @DepartamentoID = SCOPE_IDENTITY();
        end

        --inserta cliente
    
```



```

        insert into Cliente
        (NombreCliente,ApellidoCliente,Telefono,Direccion,DepartamentoID,CodigoPostal
        )
        values (@NombreCliente, @ApellidoCliente, @Telefono,
        @Direccion, @DepartamentoID, @CodigoPostal);

        commit transaction;
        print 'Cliente insertado correctamente';
    end try
    begin catch
        rollback transaction;

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
        DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
        DECLARE @ErrorState INT = ERROR_STATE();

        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    end catch
end;
go

```

Ejecución

```

exec sp_InsertarCliente 'Sarah Daniela Fernanda', 'Herrera Arias', 'Los
Gladiolos 666', 'Cajamarca', 'Peru', '976666666', '06003'
exec sp_InsertarCliente 'Jhonatan', 'Quispe', 'Hell 666', 'La Libertad',
'Peru', '976566666', '13001'
exec sp_InsertarCliente 'Andree', 'Anthonelloncio', 'Four 666', 'Antioquia',
'Colombia', '+57 316 1121000', '05001'

```

ClienteID	NombreCliente	ApellidoCliente	Telefono	Direccion	DepartamentoID	CodigoPostal
1	Anthonela	Lima	NULL		2	
2	Arnold	Ocas	NULL		2	
3	Arturo	Valdiviezo	NULL		2	
4	Darick	Pérez	NULL		2	
5	Sarah	Herrera	NULL		2	
6	Sarah Daniela Fernanda	Herrera Arias	976666666	Los Gladiolos 666	2	06003
7	Jhonatan	Quispe	976566666	Hell 666	3	13001
8	Andree	Anthonelloncio	+57 316 1121000	Four 666	4	05001

b. Obtener Ventas por Cliente

```

CREATE PROCEDURE sp_ObtenerVentasPorCliente
    @ClienteID INT -- Identificador del cliente para filtrar las ventas
AS
BEGIN
    BEGIN TRY
        -- Consulta detallada de las ventas del cliente
        SELECT
            c.ClienteID,
            CONCAT(c.NombreCliente, ' ', c.ApellidoCliente) AS
NombreCompletoCliente,
            v.VentaID,
            v.FechaVenta,
            P.NombreProducto,
            dv.Cantidad,
            dv.PrecioUnitario,
            dv.TotalBruto,
            dv.Descuento,

```

```

        dv.Total AS TotalVenta
FROM
    Venta v
INNER JOIN
    Cliente c ON v.ClienteID = c.ClienteID
INNER JOIN
    DetalleVenta dv ON v.VentaID = dv.VentaID
INNER JOIN
    Producto p ON dv.ProductoID = p.ProductoID
WHERE
    c.ClienteID = @ClienteID
ORDER BY
    v.FechaVenta DESC; -- Ordena las ventas por fecha de manera
descendente
END TRY
BEGIN CATCH
    -- Manejo de errores
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
    DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
    DECLARE @ErrorState INT = ERROR_STATE();

    RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
END CATCH
END;
GO

```

Ejecución

```
exec sp_ObtenerVentasPorCliente 1
```

CientelID	NombreCompletoCliente	VentaID	FechaVenta	NombreProducto	Cantidad	PrecioUnitario	TotalBruto	Descuento	TotalVenta
1	Anthoneia Limay	10	2024-12-10 00:00:00.000	Mani tostado bola	0.25	17	4.25	0.00	4.25
1	Anthoneia Limay	10	2024-12-10 00:00:00.000	Habas tostadas	0.50	17	8.50	0.00	8.50
1	Anthoneia Limay	10	2024-12-10 00:00:00.000	Flor de Jamaica	0.25	22	5.50	0.00	5.50
1	Anthoneia Limay	8	2024-12-08 00:00:00.000	Tomillo molido	0.10	15	1.50	0.00	1.50
1	Anthoneia Limay	8	2024-12-08 00:00:00.000	Semillas de linaza	0.10	12	1.20	0.00	1.20
1	Anthoneia Limay	8	2024-12-08 00:00:00.000	Semilla de chia	0.10	20	2.00	0.00	2.00
1	Anthoneia Limay	8	2024-12-08 00:00:00.000	Canela en polvo	0.50	30	15.00	0.00	15.00
1	Anthoneia Limay	8	2024-12-08 00:00:00.000	Comino molido	0.10	15	1.50	0.00	1.50

c. Cambiar Precio del Producto

```

CREATE PROCEDURE sp_cambiarPrecioUnitario
    @ProductoID INT,
    @PrecioNuevo MONEY
AS
BEGIN
    UPDATE Producto
    SET PrecioUnitario = @PrecioNuevo
    WHERE ProductoID = @ProductoID
END

```

Ejecución

	ProductoID	NombreProducto	CategorialID	ProveedorID	PrecioUnitario	UnidadesEnStock	Descontinuado
1	1	Almendras naturales	1	1	44,00	4.50	0
2	2	Almendras tostadas	1	1	46,00	5.00	0
3	3	Castañas de primera	1	1	48,00	6.00	0
4	4	Cashew tostados	1	1	44,00	7.00	0

```
EXEC sp_cambiarPrecioUnitario 1, 41
```

	ProductoID	NombreProducto	CategorialID	ProveedorID	PrecioUnitario	UnidadesEnStock	Descontinuado
1	1	Almendras naturales	1	1	41,00	4.50	0

d. Ingresar detalle de venta

```

ALTER PROCEDURE sp_AgregarDetalleVenta
    @NombreCliente NVARCHAR(100),
    @ApellidoCliente NVARCHAR(100),
    @EmpleadoID INT,
    @VentaID INT,
    @ProductoID INT,
    @Cantidad DECIMAL(10,2)
AS
    BEGIN
        BEGIN TRY
            BEGIN TRANSACTION;
            DECLARE @ClienteID INT;
            -- Intentamos obtener el ClienteID
            SELECT @ClienteID = ClienteID
            FROM Cliente
            WHERE NombreCliente = @NombreCliente
                AND ApellidoCliente =
@ApellidoCliente;

            IF @ClienteID IS NULL
            BEGIN
                -- insertando nuevo cliente
                EXEC
sp_InsertarSoloNombreYApellidoCliente @NombreCliente, @ApellidoCliente;
                -- reaccinando clienteID
                SELECT @ClienteID = ClienteID
                FROM Cliente
                WHERE NombreCliente = @NombreCliente
                    AND ApellidoCliente =
@ApellidoCliente;
            END;

            -- Intentamos obtener el ID de la venta
            DECLARE @VentaIDAuxiliar INT;

            IF @VentaID = (SELECT MAX(VentaID) + 1 FROM
Venta)
            BEGIN
                -- SI VENTAID ES MAYOR A LA ULTIMA VENTA, SE
                AGREGA LA NUEVA VENTA
                EXEC sp_ingresarNuevaVenta @ClienteID,
@EmpleadoID;
                PRINT 'Venta insertada correctamente.';

                -- reaccinando ventaID
                SELECT @VentaIDAuxiliar = MAX(VentaID)
                FROM Venta
                WHERE ClienteID = @ClienteID;
            END;
            -- SI VENTAID ES MAYOR A LA ULTIMA VENTA O
            SIGUIENTE, SE CANCELA LA OPERACIÓN
        
```

```

Venta)
    IF @VentaID > (SELECT MAX(VentaID) + 1 FROM
BEGIN
    ROLLBACK TRANSACTION;
    PRINT 'La venta no se puede generar,
conflicto con VentaID.';
    RETURN;
END;

-- SI VENTAID ES MENOR A LA ULTIMA VENTA, SE
CANCELA LA OPERACIÓN
Venta))
    IF @VentaID < ((SELECT MAX(VentaID) FROM
BEGIN
    ROLLBACK TRANSACTION;
    PRINT 'No se puede ingresar un
producto en una venta anterior.';
    RETURN;
END;

-- SI VENTAID ES IGUAL A LA ULTIMA VENTA, SE
AGREGA EL NUEVO PRODUCTO
Venta)
    IF @VentaID = (SELECT MAX(VentaID) FROM
BEGIN
    SET @VentaIDAuxiliar = @VentaID;
END
    -- CALCULAMOS EL DESCUENTO
    DECLARE @Descuento DECIMAL (10,2);
    SET @Descuento =
dbo.fn_CalcularDescuento(@Cantidad);

    -- INSERTAMOS UN NUEVO PRODUCTO EN EL DETALLE
DE VENTA
    INSERT INTO DetalleVenta (VentaID,
ProductoID, Cantidad, Descuento)
VALUES (@VentaIDAuxiliar, @ProductoID,
@Cantidad, @Descuento);

    COMMIT TRANSACTION;
    PRINT 'Detalle de venta insertada
correctamente.';

END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;

    -- Mostrar el mensaje de error
    DECLARE @ErrorMessage NVARCHAR(4000) =
ERROR_MESSAGE();

    PRINT 'Error: ' + @ErrorMessage;
    RAISERROR (@ErrorMessage, 16, 1);
END CATCH
END;

```

Ejecución

```
EXEC sp_AgregarDetalleVenta 'Anthonela', 'Limay', 1, 11, 70, 1
```

VentaID	ProductoID	PrecioUnitario	Cantidad	TotalBruto	Descuento	Total
11	62	72	1.00	72.00	0.00	72.00
11	70	7	1.00	7.00	0.00	7.00

e. Ingresar solo Nombre y Apellido de un Cliente

```
CREATE PROCEDURE sp_InsertarSoloNombreYApellidoCliente
@Nombre NVARCHAR(50),
@Apellido NVARCHAR(50)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;
        INSERT INTO Cliente(NombreCliente, ApellidoCliente)
        VALUES (@Nombre, @Apellido);

        COMMIT TRANSACTION;
        PRINT 'Cliente insertado correctamente';
    END TRY

    BEGIN CATCH
        ROLLBACK TRANSACTION;
    END CATCH
END
```

Ejecucion

```
EXEC sp_InsertarSoloNombreYApellidoCliente 'Pepe', 'Marin'
```

ClienteID	NombreCliente	ApellidoCliente	Telefono	Direccion	DepartamentoID	CodigoPostal
8	Andree	Anthonelloncio	+57 31...	Four 666	4	05001
9	pepe	marin	92361...	NULL	NULL	NULL

f. Ingresar Nueva Venta

```
CREATE PROCEDURE sp_ingresarNuevaVenta
@ClienteID INT,
@EmpleadoID INT
AS
BEGIN
    INSERT INTO Venta (ClienteID, EmpleadoID, FechaVenta)
    VALUES (@ClienteID, @EmpleadoID, GETDATE())
END
```

Ejecución

```
EXEC sp_ingresarNuevaVenta 5, 1
```

VentaID	CienteID	EmpleadoID	FechaVenta
12	5	1	2025-01-25 11:45:42.447

g. Ingresar nuevo empleado

```
CREATE PROCEDURE sp_ingresarNuevoEmpleado
    @NombreEmpleado NVARCHAR(15),
    @ApellidoEmpleado NVARCHAR(15),
    @FechaContrato DATETIME,
    @Telefono NVARCHAR(15),
    @Direccion NVARCHAR(30),
    @FechaNacimiento DATETIME,
    @DepartamentoID INT,
    @EstadoEmpleado INT
AS
BEGIN
    INSERT INTO Empleado(NombreEmpleado, ApellidoEmpleado,
        FechaContrato, Telefono, Direccion, FechaNacimiento, DepartamentoID,
        EstadoEmpleado)
        VALUES (@NombreEmpleado, @ApellidoEmpleado, @FechaContrato,
            @Telefono, @Direccion, @FechaNacimiento, @DepartamentoID,
            @EstadoEmpleado)
END
```

Ejecución

```
EXEC sp_ingresarNuevoEmpleado 'Cecilia', 'Sánchez', NULL, NULL, NULL,
    NULL, NULL, 3
```

EmpleadoID	NombreEmpleado	ApellidoEmpleado	FechaContrato	Telefono	Direccion	FechaNacimiento	DepartamentoID	EstadoEmpleado
1	Pablo	Herrera	2024-03-25 00:00:00.000	955967803	Jr. Juncos con Gladiolos	1987-01-07 00:00:00.000	2	1
2	Camila	Herrera	2024-03-26 00:00:00.000		Jr. Tulipanes	2007-04-04 00:00:00.000	2	1
3	Cecilia	Sánchez	NULL	NULL	NULL	NULL	NULL	3

4) Funciones

a. Calcular TotalBruto

```
CREATE FUNCTION fn_TotalBruto (@PrecioUnitario MONEY, @cantidad DECIMAL(10,
    2))
    RETURNS DECIMAL(10, 2)
AS
BEGIN
    RETURN @PrecioUnitario * @Cantidad;
END;
```

b. Calcular Total

```
CREATE FUNCTION fn_Total (@TotalBruto DECIMAL(10, 2), @descuento
    DECIMAL(10, 2))
    RETURNS DECIMAL(10, 2)
AS
BEGIN
```

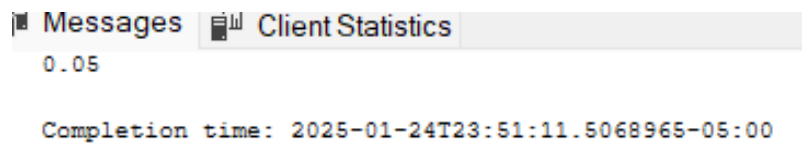
```
RETURN @TotalBruto * (1 - @descuento);
END;
```

c. *Calcular Descuento*

```
Alter FUNCTION fn_CalcularDescuento(@Cantidad DECIMAL(10, 2))
RETURNS DECIMAL(5, 2) -- El valor que devuelve es el porcentaje de descuento
AS
BEGIN
    DECLARE @Descuento DECIMAL(5, 2);

    IF @Cantidad >= 20
    BEGIN
        SET @Descuento = 0.2; -- 20% de descuento si la cantidad es mayor o
        igual a 20 kilos
    END
    ELSE IF @Cantidad >= 5
    BEGIN
        SET @Descuento = 0.05; -- 5% de descuento si la cantidad es mayor o
        igual a 5 kilos
    END
    ELSE
    BEGIN
        SET @Descuento = 0; -- No hay descuento si la cantidad es menor a 5
        kilos
    END

    RETURN @Descuento; -- Retornamos el valor de descuento calculado
END;
```



Messages Client Statistics

0.05

Completion time: 2025-01-24T23:51:11.5068965-05:00

d. *Nombre Completo de Cliente*

```
CREATE FUNCTION dbo.GetNombreCompletoCliente (@ClienteID INT)
RETURNS NVARCHAR(101)
AS
BEGIN
    DECLARE @NombreCompleto NVARCHAR(101)
    SELECT @NombreCompleto = NombreCliente + ' ' + ApellidoCliente
    FROM Cliente
    WHERE ClienteID = @ClienteID
    RETURN @NombreCompleto
END
```

Ejecución

```
SELECT dbo.GetNombreCompletoCliente(1) AS NombreCompletoCliente
```

	NombreCompletoCliente
1	Anthonela Limay

e. Total de una venta

```
CREATE FUNCTION dbo.GetTotalVenta (@VentaID INT)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @Total DECIMAL(10, 2)
    SELECT @Total = SUM(Total)
    FROM DetalleVenta
    WHERE VentaID = @VentaID
    RETURN @Total
END
```

Ejecución

```
SELECT dbo.GetTotalVenta(1) AS TotalVenta
```

	TotalVenta
1	44.00

f. Productos que suministra un Proveedor

```
CREATE FUNCTION dbo.GetNumeroProductosProveedor (@ProveedorID INT)
RETURNS INT
AS
BEGIN
    DECLARE @NumeroProductos INT
    SELECT @NumeroProductos = COUNT(*)
    FROM Producto
    WHERE ProveedorID = @ProveedorID
    RETURN @NumeroProductos
END
```

Ejecución

```
SELECT dbo.GetNumeroProductosProveedor(1) AS NumeroProductosProveedor
```

NumeroProductosProveedor
44

g. Cantidad de productos por categoría

```
CREATE FUNCTION dbo.GetNumeroProductosCategoria (@CategoriaID INT)
RETURNS INT
```



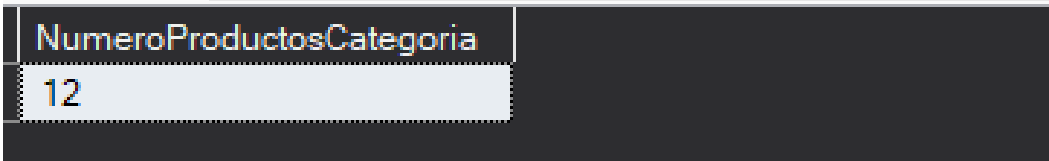
```

AS
BEGIN
    DECLARE @NumeroProductos INT
    SELECT @NumeroProductos = COUNT(*)
    FROM Producto
    WHERE CategoriaID = @CategoriaID
    RETURN @NumeroProductos
END

```

Ejecución

```
SELECT dbo.GetNumeroProductosCategoria(1) AS NumeroProductosCategoria
```



NumeroProductosCategoria
12

5) Vistas

a. Vista de ventas detalladas

```

CREATE VIEW VentasDetalladas AS
SELECT
    v.VentaID,
    c.NombreCliente,
    c.ApellidoCliente,
    e.NombreEmpleado,
    e.ApellidoEmpleado,
    v.FechaVenta,
    dv.ProductoID,
    p.NombreProducto,
    dv.Cantidad,
    dv.PrecioUnitario,
    dv.Total
FROM
    Venta v
INNER JOIN
    Cliente c ON v.ClienteID = c.ClienteID
INNER JOIN
    Empleado e ON v.EmpleadoID = e.EmpleadoID
INNER JOIN
    DetalleVenta dv ON v.VentaID = dv.VentaID
INNER JOIN
    Producto p ON dv.ProductoID = p.ProductoID;
GO

```

Ejecución

```
SELECT * FROM VentasDetalladas
```

VentaID	NombreCliente	ApellidoCliente	NombreEmpleado	ApellidoEmpleado	FechaVenta	ProductoID	NombreProducto	Cantidad	PrecioUnitario	Total
1	Anthonela	Limay	Pablo	Herrera	2024-12-01 00:00:00.000	2	Almendras tostadas	0.25	46	11.50
1	Anthonela	Limay	Pablo	Herrera	2024-12-01 00:00:00.000	5	Pistachos tostados	0.25	56	14.00
1	Anthonela	Limay	Pablo	Herrera	2024-12-01 00:00:00.000	7	Pecanas peladas	0.25	74	18.50
2	Arturo	Valdiviezo	Camila	Herrera	2024-12-02 00:00:00.000	69	Harina de almendras	0.50	40	20.00
2	Arturo	Valdiviezo	Camila	Herrera	2024-12-02 00:00:00.000	70	Harina de 7 semillas	1.00	7	7.00

b. Productos por categoría

```
CREATE VIEW ProductosPorCategoria AS
SELECT
    cp.NombreCategoria,
    p.ProductoID,
    p.NombreProducto,
    p.PrecioUnitario,
    p.UnidadesEnStock
FROM
    Producto p
INNER JOIN
    CategoriaProducto cp ON p.CategoriaID = cp.CategoriaID
GO
```

Ejecución

```
SELECT * FROM ProductosPorCategoria
```

NombreCategoria	ProductoID	NombreProducto	PrecioUnitario	UnidadesEnStock
Frutos Secos	14	Avellanas tostadas	60,00	3.00
Frutos Secos	15	Nueces peladas	46,00	1.00
Frutos Secos	16	Nueces con cascara	22,00	7.00
Frutos Secos	17	Manteca de vaca	14,00	2.00

c. Venta por Cliente

```
CREATE VIEW ClientesVentas AS
SELECT
    c.ClienteID,
    c.NombreCliente,
    c.ApellidoCliente,
    COUNT(v.VentaID) AS TotalVentas,
    SUM(dv.Total) AS TotalPagado
FROM
    Cliente c
INNER JOIN
    Venta v ON c.ClienteID = v.ClienteID
INNER JOIN
    DetalleVenta dv ON v.VentaID = dv.VentaID
GROUP BY c.ClienteID, c.NombreCliente, c.ApellidoCliente;
GO
```

Ejecución

```
SELECT * FROM ClientesVentas
```

ClietelD	NombreCiente	ApellidoCiente	TotalVentas	TotalPagado
1	Anthonela	Limay	19	221.95
2	Arnold	Ocas	3	28.50
3	Arturo	Valdiviezo	7	97.50
4	Darick	Pérez	7	99.50

d. Cantidad de ventas por empleado

```
CREATE VIEW EmpleadosVentas AS
SELECT
    e.EmpleadoID,
    e.NombreEmpleado,
    e.ApellidoEmpleado,
    COUNT(v.VentaID) AS TotalVentas,
    SUM(dv.Total) AS TotalVendido
FROM
    Empleado e
INNER JOIN
    Venta v ON v.EmpleadoID = e.EmpleadoID
INNER JOIN
    DetalleVenta dv ON dv.VentaID = v.VentaID
GROUP BY
    e.EmpleadoID, e.NombreEmpleado, e.ApellidoEmpleado;
GO
```

Ejecución

```
SELECT * FROM EmpleadosVentas
```

EmpleadoID	NombreEmpleado	ApellidoEmpleado	TotalVentas	TotalVendido
1	Pablo	Herrera	17	223.50
2	Camila	Herrera	19	223.95

e. Producto en stock

```
CREATE VIEW ProductosEnStock AS
SELECT
    p.ProductoID,
    p.NombreProducto,
    p.UnidadesEnStock,
    cp.NombreCategoria,
    pr.NombreProveedor
FROM
    Producto p
JOIN
    CategoriaProducto cp ON p.CategoriaID = cp.CategoriaID
JOIN
    Proveedor pr ON p.ProveedorID = pr.ProveedorID
WHERE
    p.UnidadesEnStock > 0
```

GO

Ejecución

```
SELECT*FROM ProductosEnStock
```

ProductoID	NombreProducto	UnidadesEnStock	NombreCategoria	NombreProveedor
1	Almendras naturales	4.50	Frutos Secos	FrutoLar
2	Almendras tostadas	5.00	Frutos Secos	FrutoLar
3	Castañas de primera	6.00	Frutos Secos	FrutoLar
4	Cashews tostados	7.00	Frutos Secos	FrutoLar
5	Pistachos tostados	4.00	Frutos Secos	FrutoLar
6	Pecanas con cascara	5.00	Frutos Secos	FrutoLar

f. Ventas por Fecha

```
CREATE VIEW VentasFecha AS
SELECT
    v.VentaID,
    v.FechaVenta,
    c.NombreCliente,
    c.ApellidoCliente,
    SUM(dv.Total) AS TotalVenta
FROM
    Venta v
INNER JOIN
    Cliente c ON v.ClienteID = c.ClienteID
INNER JOIN
    DetalleVenta dv ON v.VentaID = dv.VentaID
GROUP BY
    v.VentaID, v.FechaVenta, c.NombreCliente, c.ApellidoCliente; --
Asegúrate de incluir todas las columnas no agregadas aquí
GO
```

Ejecución

```
SELECT*FROM VentasFecha ORDER BY FechaVenta DESC;
```

VentaID	FechaVenta	NombreCliente	ApellidoCliente	TotalVenta
1	2024-12-01 00:00:00.000	Anthonela	Limay	44.00
2	2024-12-02 00:00:00.000	Arturo	Valdiviezo	35.00
3	2024-12-03 00:00:00.000	Darick	Pérez	87.00
4	2024-12-04 00:00:00.000	Anthonela	Limay	3.00

g. Productos descontinuados

```
CREATE VIEW ProductosDescontinuados AS
SELECT
    p.ProductoID,
    p.NombreProducto,
```

```

        p.CategoriaID,
        p.PrecioUnitario,
        p.UnidadesEnStock,
        p.Descontinuado
FROM
    Producto p
WHERE
    p.Descontinuado = 1;
GO

```

Ejecución

```
SELECT * FROM ProductosDescontinuados
```

ProductoID	NombreProducto	CategoriaID	PrecioUnitario	UnidadesEnStock	Descontinuado
64	Lluvia de colores	11	30,00	2,00	1
66	Chocolate Shilico	13	8,00	10,00	1
83	Cacao en nibs	13	35,00	10,00	1
86	Huesillos	2	32,00	9,00	1
88	Caramelo de menta	2	12,00	4,00	1

h. PersonalDesempleado

```

CREATE VIEW v_empleadosDesempleados
AS
    SELECT *
    FROM Empleado
    WHERE EstadoEmpleado = '3'

```

Ejecución

```
SELECT * FROM v_empleadosDesempleados
```

EmpleadoID	Nombre...	Apellido...	Fecha...	Telefono	Direccion	Fecha...	Depart...	EstadoEmpleado
3	Cecilia	Sánchez	2025...	NULL	NULL	NULL	NULL	3

i. PersonalActivos

```

CREATE VIEW v_empleadosActivos
AS
    SELECT *
    FROM Empleado
    WHERE EstadoEmpleado = '1'

```

Ejecución

```
SELECT * FROM v_empleadosActivos
```

EmpleadoID	Nombre...	Apellido...	FechaContrato	Telefono	Direccion	FechaNacimiento	Depart...	EstadoEmpleado
1	Pablo	Herrera	2024-03-25 ...	955967803	Jr. Juncos...	1987-01-07 00:0...	2	1
2	Camila	Herrera	2024-03-26 ...		Jr. Tulipa...	2007-04-04 00:0...	2	1
4	Nicolás	Herrera	NULL	NULL	NULL	NULL	NULL	1

6) Triggers

a. Calcular Preciounitario, Total Bruto Y Total General

```
CREATE TRIGGER trg_CalcularTotales
ON DetalleVenta
AFTER INSERT
AS
    BEGIN
        UPDATE DetalleVenta
        SET DetalleVenta.PrecioUnitario = p.PrecioUnitario,
            TotalBruto = dbo.fn_TotalBruto(p.PrecioUnitario, i.Cantidad),
            Total =
            dbo.fn_Total(dbo.fn_TotalBruto(p.PrecioUnitario, i.Cantidad),
            (i.Descuento))
        FROM DetalleVenta dv INNER JOIN inserted i ON
        dv.VentaID = i.VentaID AND dv.ProductoID = i.ProductoID
        INNER JOIN Producto p ON p.ProductoID =
        i.ProductoID;
    END

    SELECT * FROM DetalleVenta

    INSERT INTO DetalleVenta (VentaID, ProductoID, Cantidad)
    VALUES (4, 1, 0.5);
```

Ejecución

```
INSERT INTO DetalleVenta (VentaID, ProductoID, Cantidad)
VALUES (11, 1, 0.5);

SELECT * FROM DetalleVenta
```

31	11	41	30	0.50	15.00	0.00	15.00
32	11	36	15	0.10	1.50	0.00	1.50
33	11	1	44	0.50	22.00	0.00	22.00

b. Denegar Eliminaciones Y Actualizaciones

```
CREATE TRIGGER trg_actualizarPrecio
ON DetalleVenta
INSTEAD OF DELETE, UPDATE
AS
    BEGIN
        ROLLBACK TRANSACTION;
        RAISERROR('No se permite eliminar ni actualizar
registros en la tabla DetalleVenta.', 16, 1);
    END
```

Ejecución

```
UPDATE DetalleVenta
SET Cantidad = 3
WHERE VentaID = 4 AND productoID = 1
```

```
Mensaje 50000, nivel 16, estado 1, procedimiento trg_actualizarPrecio, lí
No se permite eliminar ni actualizar registros en la tabla DetalleVenta.

(0 filas afectadas)

Hora de finalización: 2025-01-24T18:56:01.8523330-05:00
```

```
DELETE DetalleVenta WHERE VentaID = 4 AND productoID = 76
```

```
Mensaje 50000, nivel 16, estado 1, procedimiento trg_actualizarPrecio, lí
No se permite eliminar ni actualizar registros en la tabla DetalleVenta.

(0 filas afectadas)

Hora de finalización: 2025-01-24T18:56:01.8523330-05:00
```

c. Ingresar Compra En La Última Venta

```
CREATE TRIGGER trg_insertarUltimaVenta
ON detalleVenta
AFTER INSERT
AS
BEGIN
    IF EXISTS ( SELECT * FROM inserted i WHERE i.VentaID
< (SELECT MAX(VentaID) FROM Venta) )
    BEGIN
        ROLLBACK TRANSACTION;
        RAISERROR('No se permite insertar un
producto a ventas anteriores a la última.', 16, 1);
    END
END
```

Ejecución

```
INSERT INTO DetalleVenta(VentaID, ProductoID, Cantidad)
VALUES (4, 13, 0.5)
```

```
(1 fila afectada)
Mensaje 50000, nivel 16, estado 1, procedimiento trg_insertarUltimaV
No se permite insertar un producto a ventas anteriores a la última.
Mens. 3609, Nivel 16, Estado 1, Línea 68
La transacción terminó en el desencadenador. Se anuló el lote.
```

d. Auditoria Para Actualizaciones De Los Precios De Los Productos

```
ALTER TRIGGER trg_preciosActualizados
ON Producto
AFTER UPDATE
AS
    BEGIN
        IF UPDATE(PrecioUnitario)
            BEGIN
                INSERT INTO
                HistorialPrecioProductosActualizados(ProductoID, NombreProducto,
                ProveedorID, PrecioUnitarioAnterior, PrecioUnitarioActualizado,
                FechaCambio)
                SELECT d.ProductoID, d.NombreProducto,
                d.ProveedorID, d.PrecioUnitario, i.PrecioUnitario, GETDATE()
                from inserted i inner join deleted d on
                i.ProductoID = d.ProductoID
            END
        END
```

Tabla De Auditoria

```
CREATE TABLE HistorialPrecioProductosActualizados(
    PrecioActualizadoID INT PRIMARY KEY IDENTITY(1,1) NOT NULL,
    ProductoID INT NOT NULL,
    NombreProducto NVARCHAR(50) NOT NULL,
    ProveedorID INT,
    PrecioUnitarioAnterior MONEY,
    PrecioUnitarioActualizado MONEY,
    FechaCambio DATETIME
)
```

Ejecución

	ProductoID	NombreProducto	CategorialID	ProveedorID	PrecioUnitario	UnidadesEnStock	Descontinuado
1	1	Almendras naturales	1	1	44,00	4.50	0
2	2	Almendras tostadas	1	1	46,00	5.00	0
3	3	Castañas de primera	1	1	48,00	6.00	0
4	4	Cajeta de leche	1	1	44,00	7.00	0

```
EXEC sp_cambiarPrecioUnitario 1, 41
SELECT * FROM HistorialPrecioProductosActualizados
```

	PrecioActualizadoID	ProductoID	NombreProducto	ProveedorID	PrecioUnitarioAnterior	PrecioUnitarioActualizado	FechaCambio
1	1	35	Pimienta entera	2	46,00	50,00	2025-01-23 21:58:41.230
2	2	35	Pimienta entera	2	50,00	46,00	2025-01-23 21:59:55.087
3	3	45	Anís estrella	1	90,00	85,00	2025-01-24 19:09:05.283
4	4	45	Anís estrella	1	85,00	90,00	2025-01-24 19:09:56.240
5	5	1	Almendras naturales	1	44,00	41,00	2025-01-24 19:11:52.807

	ProductoID	NombreProducto	CategorialID	ProveedorID	PrecioUnitario	UnidadesEnStock	Descontinuado
1	1	Almendras naturales	1	1	41,00	4.50	0

e. Actualizar Stock En Cada Venta

```
CREATE TRIGGER trg_actualizarStock
ON DetalleVenta
AFTER INSERT
AS
```



```

BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;
        -- Actualizamos el stock de los productos involucrados en la
venta
        UPDATE Producto
        SET UnidadesEnStock = UnidadesEnStock - i.Cantidad
        FROM Producto p INNER JOIN inserted i ON p.ProductoID =
i.ProductoID;

        -- Validación: si el stock de algún producto cae por debajo de
0, reversionamos la transacción
        IF EXISTS ( SELECT * FROM Producto WHERE UnidadesEnStock < 0 )
        BEGIN
            ROLLBACK TRANSACTION;
            PRINT 'No hay stock.';
            RETURN;
        END;

        COMMIT TRANSACTION;
        PRINT 'Stock actualizado correctamente.';
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;

        -- Mostrar el mensaje de error
        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
        RAISERROR (@ErrorMessage, 16, 1);
    END CATCH
END

```

Ejecución

ProductoID	NombreProducto	CategorialID	ProveedorID	PrecioUnitario	UnidadesEnStock	Descontinuado
25	Sacha inchi tostado	9	2	36,00	9.00	0
26	Ajonjolí tostado imp	9	2	16,00	10.00	0

```
EXEC sp_AgregarDetalleVenta 'Carlos', 'Alvarado', 1, 11, 25, 0.5
```

ProductoID	NombreProducto	CategorialID	ProveedorID	PrecioUnitario	UnidadesEnStock	Descontinuado
25	Sacha inchi tostado	9	2	36,00	8.50	0
26	Ajonjolí tostado imp	9	2	16,00	10.00	0

f. Eliminar Producto

```

CREATE TRIGGER trg_eliminarProducto
ON Producto
INSTEAD OF DELETE
AS
    BEGIN
        ROLLBACK TRANSACTION;
        --PRINT 'No se pueden eliminar ni actualizar los
detalles de venta una vez procesados.'
        RAISERROR('No se permite eliminar un producto, solo
se puede modificar le estado a "Descontinuado (1)".', 16, 1);
    END

```

```
END
```

Ejecución

```
DELETE Producto WHERE ProductoID = 117
```

```
Mensaje 50000, nivel 16, estado 1, procedimiento trg_eliminarProducto, línea 8 [línea de ir
No se permite eliminar un producto, solo se puede modificar le estado a "Des
Mens. 3609, Nivel 16, Estado 1, Línea 263
La transacción terminó en el desencadenador. Se anuló el lote.
```

g. Eliminar Empleado

```
CREATE TRIGGER trg_eliminarEmpleado
ON Empleado
INSTEAD OF DELETE
AS
BEGIN
    ROLLBACK TRANSACTION;
    --PRINT 'No se pueden eliminar ni actualizar los
    detalles de venta una vez procesados.'
    RAISERROR('No se permite eliminar un Empleado, solo
    se puede modificar le estado a "Desempleado (3)".', 16, 1);
END
```

Ejecución

```
DELETE Empleado WHERE EmpleadoID = 4
```

```
Mensaje 50000, nivel 16, estado 1, procedimiento trg_eliminarEmpleado, línea 8 [línea de ir
No se permite eliminar un Empleado, solo se puede modificar le estado a "Desempleado (3)".
Mens. 3609, Nivel 16, Estado 1, Línea 166
La transacción terminó en el desencadenador. Se anuló el lote.
```

h. Auditoría para registrar los cambios de estado de un cliente

```
CREATE TRIGGER trg_cambioEstadoEmpleado
ON Empleado
AFTER UPDATE
AS
BEGIN
    IF UPDATE(EstadoEmpleado)
    BEGIN
        -- Insertar registros en la tabla de
        auditoría
        INSERT INTO CambioEstadoEmpleado (EmpleadoID,
        NombreEmpleado, ApellidoEmpleado, EstadoAnterior, EstadoActual,
        FechaCambio)
        SELECT
            d.EmpleadoID,
            d.NombreEmpleado,
            d.ApellidoEmpleado,
            d.EstadoEmpleado AS EstadoAnterior,
```

```

        i.EstadoEmpleado AS EstadoActual,
        GETDATE()
    FROM DELETED d INNER JOIN INSERTED i ON d.EmpleadoID
    = i.EmpleadoID
    WHERE d.EstadoEmpleado != i.EstadoEmpleado;
END
END

```

Tabla de auditoría

```

CREATE TABLE CambioEstadoEmpleado (
    CambioEstadoID INT IDENTITY(1,1) PRIMARY KEY,
    EmpleadoID INT,
    NombreEmpleado NVARCHAR(50),
    ApellidoEmpleado NVARCHAR(50),
    EstadoAnterior INT,
    EstadoActual INT,
    FechaCambio DATETIME DEFAULT GETDATE()
)

```

Ejecución

```

UPDATE Empleado
SET EstadoEmpleado = 2
WHERE EmpleadoID = 2

```

CambioEstadoID	EmpleadoID	NombreEmpleado	ApellidoEmpleado	EstadoAnterior	EstadoActual	FechaCambio
1	2	Camila	Herrera	1	2	2025-01-26 12:18:42.763

7) Roles de Usuario

Existen dos tipos de usuarios que pueden manejar únicamente la base de datos “Comercial Rita”. Tenemos al administrador y los empleados. Cada uno con sus respectivos permisos de manejo.

a. Administrador

Controla la base de datos completa.

b. Empleado

Solo se le permite insertar datos en las tablas y leerlos, más no puede realizar otras actividades.

c. Script para implementar Roles de usuario

```

--ADMINISTRADOR:
-- Crear el Login (a nivel de servidor)
CREATE LOGIN LoginAdmin WITH PASSWORD = 'Admin123';

-- Crear el Usuario en la Base de Datos (a nivel de base de datos)
USE ComercialRita; -- Reemplaza con el nombre de tu base de datos
GO

```

```

CREATE USER AdminU FOR LOGIN LoginAdmin;

-- Crear el Rol Administrador en la Base de Datos
CREATE ROLE Administrador;

-- Agregar el Usuario al Rol Administrador
ALTER ROLE Administrador ADD MEMBER AdminU;

-- Otorgar TODOS los permisos al Rol Administrador
GRANT CONTROL TO Administrador;

-- Verificar la Configuración
-- Ver usuarios asignados al rol
SELECT r.name AS Rol, m.name AS Miembro
FROM sys.database_principals r
INNER JOIN sys.database_role_members rm ON r.principal_id =
rm.role_principal_id
INNER JOIN sys.database_principals m ON rm.member_principal_id =
m.principal_id
WHERE r.name = 'Administrador';

-- Ver permisos asignados al rol
SELECT permission_name, state_desc, class_desc
FROM sys.database_permissions
WHERE grantee_principal_id = DATABASE_PRINCIPAL_ID('Administrador');

--EMPLEADOS:
-- Crear los Logins (a nivel de servidor)
CREATE LOGIN LoginEmpleado1 WITH PASSWORD = 'Empleado1';
CREATE LOGIN LoginEmpleado2 WITH PASSWORD = 'Empleado2';
CREATE LOGIN LoginEmpleado3 WITH PASSWORD = 'Empleado3';

-- Crear Usuarios en la Base de Datos (a nivel de base de datos)
USE [ComercialRita]; -- Reemplaza con el nombre de tu base de datos
GO

CREATE USER EmpleadoU1 FOR LOGIN LoginEmpleado1;
CREATE USER EmpleadoU2 FOR LOGIN LoginEmpleado2;
CREATE USER EmpleadoU3 FOR LOGIN LoginEmpleado3;

-- Crear el Rol en la Base de Datos
CREATE ROLE EmpleadoR1;

-- Agregar Usuarios al Rol
ALTER ROLE EmpleadoR1 ADD MEMBER EmpleadoU1;
ALTER ROLE EmpleadoR1 ADD MEMBER EmpleadoU2;
ALTER ROLE EmpleadoR1 ADD MEMBER EmpleadoU3;

-- Otorgar permisos al Rol
-- Permiso para conectar a la base de datos
GRANT CONNECT TO EmpleadoR1;

-- Permiso para insertar en todas las tablas de la base de datos
GRANT INSERT TO EmpleadoR1;

```

```
-- Permiso para seleccionar todas las tablas y vistas de la base de
datos
GRANT SELECT TO EmpleadoR1;

-- Verificar la configuración
-- Ver usuarios asignados al rol
SELECT r.name AS Rol, m.name AS Miembro
FROM sys.database_principals r
INNER JOIN sys.database_role_members rm ON r.principal_id =
rm.role_principal_id
INNER JOIN sys.database_principals m ON rm.member_principal_id =
m.principal_id
WHERE r.name = 'EmpleadoR1';

-- Ver permisos asignados al rol
SELECT permission_name, state_desc, class_desc
FROM sys.database_permissions
WHERE grantee_principal_id = DATABASE_PRINCIPAL_ID('EmpleadoR1');
```

8) Script completo de la Base de Datos “Comercial Rita”

Script para la creación total de la base de datos “ComercialRita”.

C. Automatización de Backups

Para la creación de Backups de la Base de datos, se han considerado tres tipos de Backups con la siguiente frecuencia de realización:

1) Backup Completo

Se generan copias de seguridad completas cada mes.

Script SQL

```
--Backup completo
DECLARE @BackupFileName NVARCHAR(225);
DECLARE @FechaHora NVARCHAR(20);
SET @FechaHora = FORMAT (GETDATE(), 'yyyyMMdd_HH:mm:ss')
SET @BackupFileName = 'C:\Backups\ComercialRita\ComercialRitaCompletoMensual_'
+ @FechaHora + '.bak';
BACKUP DATABASE ComercialRita TO DISK = @BackupFileName
WITH FORMAT, INIT, NAME = 'BackupComercialRitaCompletoMensual';
```

2) Backup Diferencial

Se generan copias de seguridad diferencial cada 24 horas.

Script SQL

```
--Backup diferencial al día
DECLARE @BackupFileName NVARCHAR(225);
DECLARE @FechaHora NVARCHAR(20);
SET @FechaHora = FORMAT(GETDATE(), 'yyyyMMdd_HH:mm:ss')
SET @BackupFileName =
'C:\Backups\ComercialRita\ComercialRitaDiferencialAlDia_' + @FechaHora +
'.bak';
BACKUP DATABASE ComercialRita TO DISK = @BackupFileName
WITH DIFFERENTIAL, INIT, NAME = 'BackupComercialRitaDiferencialAlDia';
```

3) Backup Transaccional

Se generan copias de seguridad transaccionales cada hora.

Script SQL

```
--Backup transacciones a la hora
DECLARE @BackupFileName NVARCHAR(225);
DECLARE @FechaHora NVARCHAR(20);
SET @FechaHora = FORMAT(GETDATE(), 'yyyyMMdd_HH:mm:ss');
SET @BackupFileName =
'C:\Backups\ComercialRita\ComercialRitaLogTransaccionesCadaHora_' + @FechaHora
+ '.trn';
BACKUP LOG ComercialRita
TO DISK = @BackupFileName
WITH INIT, NAME = 'Backup ComercialRita Log de Transacciones';
```

4) Guardado en Amazon S3

Pasos por realizar:

- Crear usuario en la plataforma de AWS de Amazon.
- Ir al servicio de IAM y crear un nuevo usuario.
- Darle permisos para S3, o Interfaz de línea de comandos (CLI), ir al apartado de credenciales de seguridad y crear una llave de acceso.
- Crear la llave de acceso, abrir CMD (como administrador), y ejecutar el siguiente comando: `aws configure`. Después, ingresar la llave de acceso (Access Key), luego la llave de acceso secreto (Secret Access Key) y el servidor (us-east-2), y el formato de salida (json).
- En la página de inicio de AWS, ir al servicio S3, crear Bucket, con las credenciales otorgadas, y luego crear una carpeta para almacenar los backups.
- Crear en bloc de notas los siguientes comandos que se puede usar en CMD (guardar el archivo de bloc de notas como .bat):

→ @echo off

- ::Ruta de la carpeta donde se están generando los backups
- set source=C:\Backups\ComercialRita
- ::Ruta del S3 donde se guardarán los backups
- set destination=s3://comercialrita/backups
- :: Sincroniza la carpeta con S3, subiendo solo archivos nuevos o modificados
- aws s3 sync "%source%" "%destination%" --delete

g. Para que el comando pueda ejecutarse cada hora automáticamente, se debe:

- i. Abrir el Programador de Tareas (Win + R, escribir taskschd.msc, presionar Enter).
- ii. En el panel derecho, seleccionar Crear Tarea
- iii. En la pestaña General, asignar un nombre como SyncS3_ComercialRita.
- iv. En la pestaña Desencadenadores, hacer clic en Nuevo
- v. Configurar Repetir cada 1 hora, para que se ejecute cada hora.
- vi. En la pestaña Acciones, hacer clic en Nueva
- vii. Programa o script: seleccionas el archivo .bat guardado en la ruta escogida anteriormente
- viii. Agregar argumentos:
 - ExecutionPolicy Bypass -File "C:\ruta\del\script\sync_s3.(ps1 o .bat)"
- ix. Guardar la tarea y asegurarse de que esté habilitada.* /

D. Repositorio en GitHub

- a. **Nombre del repositorio:** “sql-server-database-proyecto-Comercial-Rita”.
- b. **Link:** <https://github.com/Arturo969/sql-server-database-proyecto-Comercial-Rita.git>

VI. CONCLUSIONES

A lo largo de este proyecto, se ha logrado diseñar e implementar un sistema de gestión de base de datos empresarial utilizando SQL Server y Transact-SQL (T-SQL) para la empresa Comercial Rita. Este sistema ha sustituido el uso de hojas de cálculo ineficientes, proporcionando un manejo más eficiente de la información y mejorando la seguridad de los datos. El éxito de este proyecto se puede evaluar a través de los siguientes logros obtenidos en función de los objetivos establecidos:

Se ha creado un modelo de base de datos adaptado a las necesidades de Comercial Rita, definiendo tablas, relaciones, restricciones, índices y otros elementos clave que permiten una gestión ordenada y eficiente de la información.

Se han utilizado procedimientos almacenados, vistas, triggers y funciones que mejoran significativamente la eficiencia y seguridad de las operaciones con datos. Estas implementaciones permiten automatizar procesos y asegurar la integridad de las transacciones.

Se han establecido estrategias de seguridad que protegen la integridad y confidencialidad de la información.

Se ha desarrollado y puesto en marcha un sistema automatizado que realiza respaldos periódicos de la base de datos y los almacena en Amazon S3. Este sistema garantiza la disponibilidad y recuperación de datos en caso de fallos, asegurando la continuidad del negocio.

Se han aplicado mejores prácticas para optimizar la velocidad de consulta y la eficiencia de la base de datos mediante la creación de índices y la optimización de consultas SQL. Esto ha resultado en un rendimiento mejorado y tiempos de respuesta más rápidos.

VII. RECOMENDACIONES

Se recomienda la revisión y actualización constante del sistema para mejorar su rendimiento y adaptabilidad a las necesidades cambiantes. La implementación de índices en las tablas más consultadas es esencial para optimizar el rendimiento de las consultas, especialmente a medida que la cantidad de datos crezca.

Es crucial aumentar la seguridad del sistema mediante la implementación de cifrado para proteger datos sensibles, como la información de clientes y transacciones. Además, se debe configurar auditorías en la base de datos para registrar accesos y modificaciones, estableciendo alertas para detectar actividades sospechosas.

La estrategia de backups debe ser continuamente desarrollada y mejorada, con un enfoque en backups incrementales para minimizar la pérdida de datos.

Por último, se recomienda implementar una interfaz visual que sea intuitiva y fácil de usar para los empleados de la empresa, mejorando así la eficiencia operativa y la experiencia del usuario.

VIII. REFERENCIAS

- [1] AulaClic, “Curso gratis de SQL Server. aulaClic. 2 - Introducción al SQL. Transact-SQL”. Accedido: 20 de enero de 2025. [En línea]. Disponible en: https://www.aulaclic.es/sqlserver/t_2_1.htm
- [2] J. I. O. Aznar, “Comandos útiles de Transact-SQL para la gestión de SQL Server”, JOTELULU | Servicios Cloud para Distribuidores de IT. Accedido: 20 de enero de 2025. [En línea]. Disponible en: <https://jotelulu.com/blog/comandos-utiles-de-transact-sql-para-la-gestion-de-sql-server/>
- [3] Microsoft Learn, “Introducción a Transact-SQL – Training”. Accedido: 20 de enero de 2025. [En línea]. Disponible en: <https://learn.microsoft.com/es-es/training/modules/introduction-to-transact-sql/>
- [4] Microsoft Learn, “Introducción a la programación con Transact-SQL – Training”. Accedido: 20 de enero de 2025. [En línea]. Disponible en: <https://learn.microsoft.com/es-es/training/modules/get-started-transact-sql-programming/>
- [5] EcuRed, “Transact SQL – EcuRed”. Accedido: 20 de enero de 2025. [En línea]. Disponible en: https://www.ecured.cu/Transact_SQL
- [6] TechTarget, “What is T-SQL (Transact-SQL)? | Definition from TechTarget”, Search Data Management. Accedido: 20 de enero de 2025. [En línea]. Disponible en: <https://www.techtarget.com/searchdatamanagement/definition/T-SQL>
- [7] EDteam, «¿Cómo diseñar una base de datos? ¿Cuáles son las etapas?,» 13 mayor 2020. [En línea]. Available: https://www.youtube.com/watch?v=WU1tUV_krtA&ab_channel=EDteam. [Último acceso: 08 enero 2025].
- [8] Microsoft, «Conceptos básicos del diseño de una base de datos,» [En línea]. Available: <https://support.microsoft.com/es-es/topic/conceptos-b%C3%A1sicos-del-dise%C3%B1o-de-una-base-de-datos-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5#bmdesignprocess>. [Último acceso: 18 enero 2025].
- [9] R. W. Moreira Centeno, E. E. Almeida Zambrano, H. R. Mendoza Rodríguez, E. M. San Andrés Laz y K. T. Mendoza Muñoz, Análisis y diseño de base de datos, 1 ed., vol. 1, Manta: Uleam Editorial Universitaria, 2022, p. 92.

- [10] Talent Garden, «La diferencia entre el modelo de datos conceptual y lógico,» 09 agosto 2023. [En línea]. Available: <https://blog.talentgarden.com/es/blog/data/diferencia-entre-modelo-conceptual-y-modelo-logico-datos>. [Último acceso: 19 enero 2025].
- [11] Tecnologías Información, «Modelos de datos: Modelo Conceptual, Físico y Lógico,» 2018. [En línea]. Available: <https://www.tecnologias-informacion.com/modelos-datos.html>. [Último acceso: 19 enero 2025].
- [12] LinkedIn, «¿Cuáles son las mejores metodologías de diseño de bases de datos?,» [En línea]. Available: <https://es.linkedin.com/advice/0/what-best-database-design-methodologies?lang=es>. [Último acceso: 19 enero 2025].
- [13] IBM, «¿Qué es la seguridad de la base de datos?,» [En línea]. Available: <https://www.ibm.com/mx-es/topics/database-security>. [Último acceso: 2025].
- [14] Microsoft System Center, «Implementación de roles de usuario,» 01 11 2024. [En línea]. Available: <https://learn.microsoft.com/es-es/system-center/scom/manage-security-overview?view=sc-om-2025>. [Último acceso: 2025].
- [15] Microsoft Learn, «Procedimientos almacenados (motor de base de datos),» 02 enero 2025. [En línea]. Available: <https://learn.microsoft.com/es-es/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver16#benefits-of-using-stored-procedures>.
- [16] Microsoft Learn, «Vistas,» 01 enero 2025. [En línea]. Available: <https://learn.microsoft.com/es-es/sql/relational-databases/views/views?view=sql-server-ver16>.
- [17] Microsoft Learn, «Funciones definidas por el usuario,» 02 enero 2025. [En línea]. Available: <https://learn.microsoft.com/es-es/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver16>. [Último acceso: 2025].
- [18] Microsoft Learn, «CREATE TRIGGER (Transact-SQL),» 02 enero 2025. [En línea]. Available: <https://learn.microsoft.com/es-es/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver16>. [Último acceso: 2025].
- [19] M. Mauricio, "Políticas de backup básicas que toda empresa debería tener" 17 de marzo de 2023. [En línea]. Disponible en: <https://www.dongee.com/tutoriales/politicas-de-backup-basicas-que-toda-empresa-deberia-tener-2/>. [Accedido: 19-ene-2025].

- [20] IBM, "*Políticas de copia de seguridad*" 6 de diciembre de 2024. [En línea]. Disponible en: <https://www.ibm.com/docs/es/aix/7.2?topic=concepts-backup-policies>. [Accedido: 19-ene-2025].
- [21] W. Quiroga, N. Quiroga, y WalySoft Sistemas, "*La disciplina del backup*" [En línea]. Disponible en: <https://www.walysoft.com.ar/blog/la-disciplina-del-backup.php>. [Accedido: 19-ene-2025].
- [22] Dropbox, "*Almacenamiento en la nube y almacenamiento de archivos*," Dropbox. [En línea]. Disponible en: <https://www.dropbox.com/es/features/cloud-storage>. [Accedido: 19-ene-2025].
- [23] Amazon Web Services (AWS), "*¿Qué es el almacenamiento en la nube?*" Amazon. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/cloud-storage/>. [Accedido: 19-ene-2025].