Segunda entrega del proyecto: Inserción de datos y funcionalidades

En esta segunda entrega del proyecto, nos enfocaremos en la inserción de datos desde archivos CSV y el desarrollo de funcionalidades clave como Vistas, Funciones, Stored Procedure y Triggers.

Estas herramientas complementarán la estructura inicial, automatizando procesos y facilitando la consulta y administración de la base de datos.

Preparación del Entorno y la Base de Datos

Estructura Base

Toda la estructura de la base de datos se encuentra en el archivo 'fecomInc_db_structure.sql'.

Inserción de Datos

Los datos con los que trabajaran se encuentran definidos en archivos CSV y se ingresan con las instrucciones definidas en el archivo 'fecomInc_db_inserts.sql'.

Inserción de Datos desde Archivos CSV: Preparación

Formato CSV

La carpeta 'fecomInc_data' del proyecto almacena los CSV.

Revisión de Archivos

Se verifico de manera manual pero mas simple que no se encuentren errores simples.

Limpieza

Mediante instrucciones se aseguro la eliminación de duplicados y corrección de formatos inconsistentes.

Inserción de Datos desde Archivos CSV: Implementación

1 Desactivar SQL Safe Updates

Use el comando: 'SET SQL_SAFE_UPDATES = 0;' Para continuar con la inserción de datos.

2 Definir Ruta de archivos

Se necesita definir una ruta en el LOAD DATA.
'SHOW VARIABLES LIKE 'secure_file_priv';'
Este comando mostrara la ruta predeterminada, almacene la carpeta 'fecomInc_data' o los archivos sueltos en esa dirección y haga los ajustes en el archivo inserts.

Vistas en MySQL

Vista 1: Información completa de clientes con su ubicación

Nombre de la vista:
`vista_clientes_ubicacion`

Vista 2: Vendedores y su ubicación detallada

Nombre de la vista:
`vista_vendedores_ubicacion`

Vista 3: Productos con volumen y peso

Nombre de la vista:
`vista_productos_detalle`

Vista 4: Clientes por rango etario

Nombre de la vista: `vista_clientes_rango_etario`

Vista 5: Clientes con tiempo de conversión desde suscripción a primera orden

Nombre de la vista: `vista_tiempo_conversion`

Vista 1: Información Completa de Clientes con Ubicación

Descripción:

Esta vista proporciona un resumen de los datos personales de los clientes junto con la información geográfica asociada a su código postal. Permite conocer en qué ciudad y país reside cada cliente, junto con su edad, género y fechas clave como su suscripción y su primera compra.

Utilidad:

Ideal para análisis de segmentación de mercado, estudios demográficos, campañas regionales de marketing y análisis de comportamiento por zona geográfica.

Vista 2: Vendedores y su ubicación detallada

Descripción:

Esta vista muestra los datos de cada vendedor (ID y nombre) junto con su localización exacta, incluyendo ciudad, país y coordenadas geográficas (latitud y longitud).

Utilidad:

Permite hacer mapas de calor de vendedores, analizar cobertura geográfica, optimizar la logística de envíos o hacer estudios de distribución regional.

Vista 3: Productos con volumen y peso

Descripción:

Esta vista extiende la información básica del producto incluyendo un cálculo automático del volumen cúbico en centímetros cúbicos. Presenta además el peso en gramos y las dimensiones del producto.

Utilidad:

Muy útil para el área de logística y envíos, ya que permite calcular espacios en depósito o costos de transporte. También sirve para detectar productos con características físicas fuera de lo común.

Vista 4: Clientes por rango etario

Descripción:

Agrupa a los clientes según su edad en rangos predefinidos y cuenta cuántos clientes hay en cada grupo.

Utilidad:

Permite hacer estudios de marketing por edad, entender cuál es el segmento etario predominante en la tienda y ajustar productos o campañas para cada rango.

Vista 5: Clientes con tiempo de conversión desde suscripción a primera orden

Descripción:

Muestra el tiempo (en días) que tarda cada cliente desde que se suscribió hasta que realizó su primera compra.

Utilidad:

Clave para analizar el comportamiento de nuevos usuarios, la eficacia de campañas de onboarding o identificar clientes inactivos. También permite tomar decisiones sobre estrategias de fidelización.

Funciones en MySQL

Función 1: Calcular volumen de un producto

Nombre de la función: `calcular_volumen_producto`

Función 2: Obtener el país de un cliente

Nombre de la función: `obtener_pais_cliente`

Función 1: Calcular volumen de un producto

Descripción:

Esta función recibe como parámetro el ID de un producto (`Product_ID`) y devuelve su **volumen cúbico** (en centímetros cúbicos), calculado a partir de sus dimensiones: largo, alto y ancho. En caso de que alguna de estas dimensiones no esté definida (`NULL`), se utiliza un valor por defecto de 0 para evitar errores en el cálculo

Utilidad:

Esta función encapsula la lógica de cálculo de volumen de forma reutilizable, permitiendo utilizarla en consultas para evaluar el tamaño físico de los productos. Es útil para logística, almacenamiento y categorización de productos según su volumen.

Función 2: Obtener el país de un cliente

Descripción:

Esta función recibe como parámetro el identificador de un cliente (`Customer_Trx_ID`) y devuelve el **nombre del país** donde vive ese cliente. Para esto, primero recupera su código postal desde la tabla `customers` y luego busca el país correspondiente en la tabla `Geolocations`.

Utilidad:

Permite obtener de forma rápida el país de residencia de un cliente sin necesidad de escribir una consulta con `JOIN` o subconsultas. Es útil en reportes, estadísticas por país, validaciones de envío o análisis geográfico.

Stored Procedures en MySQL

Stored Procedure 1: Listar productos con ordenamiento dinámico

Nombre: `sp_listar_productos_ordenados`

Stored Procedure 2: Actualizar el peso de un producto con control de valores inválidos

Nombre: `sp_actualizar_peso_producto`

Stored Procedure 1: Listar productos con ordenamiento dinámico

Descripción:

Este procedimiento permite consultar los productos de la tabla `products` y ordenarlos dinámicamente según un **campo** y una **dirección** (ascendente o descendente) elegidos por el usuario.
Hace uso de SQL dinámico (`PREPARE`, `EXECUTE`, `DEALLOCATE`) para construir la consulta al vuelo, permitiendo reutilizar el procedimiento con diferentes criterios.

Utilidad:

Es ideal para construir interfaces de usuario que permitan ordenar una tabla por distintos campos (peso, ancho, categoría, etc.) sin crear múltiples consultas distintas.

Stored Procedure 2: Actualizar el peso de un producto

Descripción:

Este procedimiento actualiza el campo `Product_Weight_Gr` de la tabla `products`, correspondiente al peso del producto. Antes de hacer la actualización, **verifica que el producto exista** y que el nuevo peso sea **mayor a cero**, evitando así errores comunes como referencias inválidas o valores negativos.

Utilidad:

Esta validación interna mejora la integridad de la base de datos y permite realizar cambios controlados de forma segura.

Triggers en MySQL

Trigger 1: Registro de fecha de modificación en tabla `orders` **cuando cambia el estado**

Trigger 2: Evitar que se inserten órdenes sin cliente asociado

Trigger 1: Registro de fecha de modificación en tabla `orders` cuando cambia el estado

Descripción:

Este trigger se ejecuta antes de actualizar una fila en la tabla `orders`. Si detecta un cambio en el campo `Order_Status`, actualiza automáticamente el campo `Order_Approved_At` con la fecha y hora actual. De esta forma, se registra la fecha del último cambio de estado sin necesidad de que la aplicación lo controle.

Utilidad:

Queremos saber cuándo una orden cambió de estado (por ejemplo, de "shipped" a "delivered"). Para eso, usamos un campo que **ya existe**: `Order_Approved_At`. Lo usaremos como campo de "última actualización".

Trigger 2: Evitar que se inserten órdenes sin cliente asociado

Descripción:

Este trigger actúa antes de insertar una nueva orden en la tabla `orders`. Su función es asegurar que cada orden tenga un cliente asignado, validando que el campo `Customer_ID` no esté vacío ni nulo. De este modo, se evita crear registros incompletos o inválidos desde la base de datos.

Utilidad:

Una orden debe estar siempre vinculada a un cliente. Este trigger evita que se inserten registros en `orders` con `Customer_ID` vacío o nulo.