Universidad de San Carlos de Guatemala

Centro Universitario de Occidente

División de Ciencias de la Ingeniería

Manual técnico, Gamer pro

Proyecto 1

Manejo e Implementación de Archivos

Ing. Christian López

Segundo semestre 2024

202031953 - Hania Mirleth Mazariegos Alonzo

Índice

Funciones SQL

insert_inventory

Propósito: Agrega un nuevo producto al inventario de una sucursal especificada.

Parámetros:

- p_branch_id: El ID de la sucursal donde se agregará el producto.
- p_product_id: El ID del producto que se agregará.
- p_amount: La cantidad del producto que se agregará.
- p_location: La ubicación del producto dentro de la sucursal.
- p_notes: Cualquier nota o comentario adicional sobre el producto.

Lógica:

1. Verifica si hay inventario existente: Si el producto ya existe en la sucursal, actualiza la ubicación e incrementa la cantidad.
2. Agrega nuevo inventario: Si el producto no existe, inserta dos filas: una con el estado 'en tienda' y otra con el estado 'en exhibición'.

transfer_inventory_to_display

Propósito: Transfiere una cantidad específica de un producto del estado "en tienda" al estado "en exhibición" dentro de una sucursal.

Parámetros:

- p_branch_id: El ID de la sucursal donde se realizará la transferencia.
- p_product_id: El ID del producto que se transferirá.
- p_amount: La cantidad del producto que se transferirá.
- p_location: La ubicación del producto en el estado "en exhibición".

Lógica:

1. Valida la existencia del producto: Comprueba si el producto existe en el estado "en tienda" para la sucursal dada.
2. Valida la cantidad suficiente: Asegura que haya suficiente producto en el estado "en tienda" para transferir.

3. Actualiza el inventario: Incrementa la cantidad y establece la ubicación para el estado "en exhibición". Disminuye la cantidad para el estado "en tienda".

get_branch_inventory

Propósito: recuperar información sobre todos los productos disponibles actualmente en una sucursal específica.

Devuelve: una tabla que contiene las siguientes columnas:

- product_id: identificador único del producto.
- product_name: nombre del producto.
- product_description: descripción del producto.
- product_price: precio unitario del producto.
- amount_on_display: cantidad del producto actualmente en exhibición.
- amount_in_store: cantidad del producto actualmente en almacen (no en exhibición).
- stock: cantidad total del producto disponible en la sucursal (suma de amount_on_display y amount_in_store).
- location_on_display: ubicación donde se muestra el producto en la sucursal
- location_in_store: ubicación donde se almacena el producto en la sucursal

Lógica:

1. Une product_mgmt.products con product_mgmt.inventories (opcional) en product_id y branch_id.
2. Utiliza COALESCE para manejar posibles valores nulos en las columnas de importe.
3. Filtra productos en función de si tienen un registro en exhibición o en tienda para la sucursal especificada y con un stock total mayor que cero.
4. Ordena los resultados por product_name en orden ascendente.

add_points

Propósito: Calcula y suma puntos a la cuenta de un cliente en función de una venta completada.

Parámetros:

- n_sale_id: El ID de la venta completada.

Lógica:

1. Recupera el NIT del cliente, el total de la venta y el nombre de la tarjeta.

2. Determina el multiplicador de puntos en función del tipo de tarjeta.
3. Calcula los puntos a sumar en función del total de la venta y el multiplicador.
4. Actualiza el saldo de puntos del cliente, si corresponde.

get_client_card_info

Propósito: recupera información sobre la tarjeta de un cliente, incluido el tipo, la fecha de actualización y el gasto total desde la última actualización.

Parámetros:

- nit: el número de NIT del cliente.

Devuelve: una tabla que contiene las siguientes columnas:

- client_nit: el número de NIT del cliente (el mismo que el parámetro de entrada).
- client_name: el nombre del cliente.
- current_card: el tipo de tarjeta actual (sales_mgmt.card_type).
- last_card_update: marca de tiempo de la última actualización de la tarjeta.
- total_spent: monto total gastado desde la última actualización de la tarjeta (valor predeterminado: 0).

Lógica:

1. Une sales_mgmt.clients con sales_mgmt.sales (opcional) en nit.
2. Utiliza COALESCE para manejar posibles valores nulos en la columna total de sales_mgmt.sales.
3. Filtra los resultados según el nit proporcionado y garantiza que la fecha de venta sea posterior a la última actualización de la tarjeta del cliente.
4. Agrupa los resultados por información del cliente y agrega el gasto total.

upgrade_card

Propósito: evalúa la elegibilidad de un cliente para una actualización de tarjeta y actualiza la información de la misma si corresponde.

Parámetros:

- _nit: el número NIT del cliente.

Lógica:

1. Recupera el tipo de tarjeta actual del cliente de sales_mgmt.clients.
2. Calcula el monto total gastado desde la última actualización de la tarjeta.
3. Determina el nivel de actualización potencial según la tarjeta actual y el total gastado:

4. Ninguno -> Común (si el total gastado es >= 10000).
5. Común -> Oro (si el total gastado es >= 20000).
6. Oro -> Platino (si el total gastado es >= 30000).
7. Platino -> Diamante (si el total gastado es >= 30000).
8. Genera una excepción si el cliente no cumple con los criterios de actualización.
9. Actualiza el tipo de tarjeta del cliente y la fecha de actualización en sales_mgmt.clients.
10. Genera un aviso que indica la actualización exitosa de la tarjeta y el nuevo nivel de la misma.

get_discount_history

Propósito: recupera un historial de ventas con descuentos aplicados dentro de un rango de fechas.

Parámetros:

- start_date: Fecha de inicio del rango de fechas (TIMESTAMP).
- end_date: Fecha de finalización del rango de fechas (TIMESTAMP).

Devuelve: Una tabla que contiene las siguientes columnas:

- branch_id: ID de la sucursal donde se realizó la venta.
- branch_name: Nombre de la sucursal.
- sale_id: Identificador único de la venta.
- client_name: Nombre del cliente asociado con la venta.
- total_discount: Suma de los descuentos temporales y basados en puntos aplicados a la venta.
- sale_date: Marca de tiempo de la venta.

Lógica:

1. Une sales_mgmt.sales con sales_mgmt.clients y branch_mgmt.branches en claves externas relevantes.
2. Filtra los resultados en función de las ventas con descuentos (temporales o en puntos) y dentro del rango de fechas especificado.
3. Calcula el descuento total sumando temp_discount y point_discount de la venta.
4. Ordena los resultados por fecha de venta en orden ascendente.

get_top_10_sales

Propósito: recupera información sobre las 10 ventas principales (por monto total) dentro de un rango de fechas.

Parámetros:

- start_date: Fecha de inicio del rango de fechas (TIMESTAMP).

- end_date: Fecha de finalización del rango de fechas (TIMESTAMP).

Devuelve: Una tabla que contiene las siguientes columnas:
- branch_id: ID de la sucursal donde se realizó la venta.
- branch_name: Nombre de la sucursal.
- sale_id: Identificador único de la venta.
- client_name: Nombre del cliente asociado con la venta.
- cashier_name: Nombre del cajero que creó la venta.
- total_discount: Suma de los descuentos temporales y basados en puntos aplicados a la venta.
- total: Monto total de la venta (incluidos los descuentos).
- sale_date: Marca de tiempo de la venta.

Lógica:
1. Une sales_mgmt.sales con branch_mgmt.branches, sales_mgmt.clients y branch_mgmt.cashiers en las claves externas relevantes.
2. Filtra los resultados según el rango de fechas especificado.
3. Calcula el descuento total sumando temp_discount y point_discount de la venta.
4. Ordena los resultados por total en orden descendente y limita la salida a las 10 primeras filas.

Triggers SQL

initialize_sale

Propósito: inicializa un nuevo registro de venta cuando se inserta una nueva fila en la tabla sales_mgmt.sales.

Activador: se ejecuta antes de insertar una nueva fila en sales_mgmt.sales.

Lógica:
1. Recupera el branch_id y el checkout_number asociados con el cajero que crea la venta.
2. Inicializa otros campos relacionados con la venta:
3. temp_discount: importe del descuento temporal (valor predeterminado 0).
4. point_discount: descuento aplicado mediante puntos de fidelidad (valor predeterminado 0).
5. total: importe total de la venta (valor predeterminado 0).
6. date: marca de tiempo de la creación de la venta (usa CURRENT_TIMESTAMP

insert_sale_item

Propósito: Valida y procesa el uso de puntos de fidelidad para una venta.

Activador: Se ejecuta antes de insertar una nueva fila en la tabla sales_mgmt.points_usage.

Lógica:

1. Comprueba el uso de puntos no válidos (0 o menos).
2. Recupera el client_nit y la tarjeta asociada a la venta.
3. Obtiene los puntos totales del cliente y el monto total de la venta.
4. Valida si los puntos a utilizar superan el total de puntos del cliente o el total de la venta.
5. Aplica el descuento de puntos al total de la venta.
6. Deduce los puntos utilizados del total de puntos del cliente.
7. Devuelve: El registro NUEVO modificado.