Proyecto Final

Segunda entrega

# Vistas

Vista 1: USUARIO\_COMPRAS

Descripción:

La vista USUARIO\_COMPRAS está diseñada para proporcionar una visualización detallada de los usuarios y sus compras individuales.

Objetivo:

El propósito principal de esta vista es permitir a los programadores visualizar y comprender fácilmente la actividad de compra de cada usuario en particular. Al mostrar los detalles de las compras de libros junto con la información del usuario, la vista facilita un análisis detallado de las preferencias de compra de los usuarios individuales

Tablas/Datos:

Esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

* id\_user: El ID único del usuario que realizó la compra.
* username: El nombre de usuario asociado al usuario en cuestión.
* first\_name: El nombre del usuario.
* last\_name: El apellido del usuario.
* id\_purchase: El ID único de la compra realizada.
* title: El título del libro.
* date\_of\_purchase: La fecha en que se realizó la compra.
* quantity: La cantidad de libros comprados.
* subtotal: El subtotal de la compra realizada.

Las tablas son:

* usuario: Contiene información detallada sobre los usuarios registrados.
* compra: Almacena detalles específicos de las compras realizadas por los usuarios, como la fecha de compra, la cantidad y el subtotal.
* publicacion: Proporciona información sobre las publicaciones específicas de libros, incluidos los precios y el stock.
* ficha\_libro: Contiene detalles exhaustivos sobre los libros disponibles.

Vista 2: INFO\_LIBRO

Descripción:

Esta vista está diseñada para proporcionar una visualización detallada un libro específico disponible en el sistema. Esta vista combina información esencial de múltiples tablas relacionadas a ficha\_libro tales como libro\_genero, libro\_autor y libro\_editorial, para ofrecer una visión completa de los detalles del mismo.

Objetivo:

El objetivo de esta vista es permitir un acceso y visualización más sencillo a la información clave de los libros disponibles. Al mostrar detalles como el título del libro, el género, el autor y la editorial, la vista facilita a los usuarios obtener una comprensión clara de la información de cada libro en particular.

Tablas/Datos:

Esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

* id\_book: El ID único del libro en la base de datos..
* title: El título del libro.
* sku: El número ISBN del libro.
* book\_description: Una descripción detallada del libro.
* genre: El género al que pertenece el libro.
* author: El autor del libro.
* publisher: La editorial que publicó el libro.

Las tablas serían:

* ficha\_libro: Contiene información detallada sobre cada libro en la base de datos, como el título, la descripción y el número ISBN del libro.
* libro\_genero: Almacena información sobre los diferentes géneros literarios a los que pertenecen los libros.
* libro\_autor: Proporciona detalles sobre los autores de los libros disponibles en la base de datos.
* libro\_editorial: Contiene información sobre las editoriales que han publicado los libros disponibles en el sistema.

Vista 3: PUBLICACION\_MENSAJES

Descripción:

Esta vista está diseñada para proporcionar una visualización integral de todos los mensajes asociados con cada publicación. Esta vista combina datos importantes de múltiples tablas, incluidas publicacion, mensaje y usuario, para ofrecer una visión detallada de los mensajes relacionados con cada publicación en particular.

Objetivo:

El objetivo principal de esta vista es permitir a los programadores visualizar y comprender fácilmente todos los mensajes asociados con cada publicación en el sistema. Al mostrar detalles como el ID de la publicación, el ID de usuario, el contenido del mensaje, la fecha del mensaje y cualquier respuesta asociada, la vista facilita un análisis completo de la interacción y la comunicación relacionadas con cada publicación.

Tablas/Datos:

Esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

* id\_publication: El ID único de la publicación asociada con el mensaje.
* id\_user: El ID único del usuario que envió el mensaje.
* first\_name: El nombre del usuario que envió el mensaje.
* content: El contenido del mensaje enviado.
* date\_of\_message: La fecha en que se envió el mensaje.
* reply: Cualquier respuesta asociada con el mensaje, si la hay.

Las tablas serían:

* publicacion: Contiene información detallada sobre las publicaciones.
* mensaje: Almacena todos los mensajes asociados con las publicaciones, incluido el contenido del mensaje, las fechas y cualquier respuesta relacionada.
* usuario: Proporciona detalles sobre los usuarios que han enviado mensajes relacionados con las publicaciones, como el nombre y el ID de usuario.

Vista 4: USUARIO\_COMPRAS\_TOTALES

Descripción:

Esta vista proporciona una visión consolidada del total de las compras realizadas por cada usuario. Esta vista combina datos clave de las tablas usuario y compra para mostrar el precio total de todas las compras realizadas por cada usuario en particular.

Objetivo:

El objetivo principal de esta vista es permitir a los programadores obtener una visión global del valor total de las compras realizadas por cada usuario en el sistema. Al mostrar el precio total final de todas las compras realizadas para cada usuario, la vista facilita un análisis rápido y eficiente del consumo de cada cliente individual.

Tablas/Datos:

Esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

* id\_user: El ID único del usuario.
* username: El nombre de usuario asociado al usuario en cuestión.
* first\_name: El nombre del usuario.
* last\_name: El apellido del usuario.
* total\_sales: El monto total de todas las compras realizadas por el usuario, calculado como la suma de los subtotales de todas las compras.

Las tablas serían:

* usuario: Contiene información detallada sobre los usuarios registrados..
* compra: Almacena detalles específicos de las compras realizadas por los usuarios, como la fecha de compra, la cantidad y el subtotal.

Vista 5: USUARIO\_CANTIDAD\_COMPRAS

Descripción:

Esta vista permite visualizar la cantidad de compras realizadas por cada usuario. Esta vista combina datos importantes de las tablas usuario y compra para mostrar la cantidad total de compras realizadas por cada usuario individual.

Objetivo:

El objetivo principal de esta vista es permitir a los programadores obtener una comprensión clara de la cantidad de compras realizadas por cada usuario en particular. Al mostrar la cantidad de compras para cada usuario, la vista facilita un análisis rápido y eficiente del comportamiento de compra de cada cliente individual.

Tablas/Datos:

Esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

* id\_user: El ID único del usuario en el sistema.
* username: El nombre de usuario asociado al usuario en cuestión.
* email: La dirección de correo electrónico del usuario.
* purchase\_count: El número total de compras realizadas por el usuario.

Las tablas son:

* usuario: Contiene información detallada sobre los usuarios registrados.
* compra: Almacena detalles específicos de las compras realizadas por los usuarios, como la fecha de compra, la cantidad y el subtotal.

1. Funciones

Función 1: STOCK\_GENERO

Descripción:

Esta funcion se utiliza para calcular la cantidad total de libros en stock de un género específico. Esta función combina datos esenciales de las tablas publicacion y ficha\_libro para determinar la cantidad total de libros disponibles en stock para un género particular.

Objetivo:

El propósito principal de esta funcion es proporcionar una manera eficiente de calcular el inventario total de libros disponibles para la venta dentro de un género específico. Al realizar este cálculo, la función permite a los usuarios obtener rápidamente una visión clara de la disponibilidad de libros dentro de un género determinado.

Tablas/Datos:

Esta funcion se compone de los siguientes elementos clave:

* Parámetro de entrada: id\_genero - Un entero que representa el ID del género para el cual se desea calcular el inventario total.
* Variable local: stock\_total - Un entero que almacena el valor total de stock para el género especificado.
* Consulta SQL: La función utiliza una consulta SQL que calcula la suma total del stock de libros para un género específico basado en el parámetro de entrada id\_genero.

Esta funcion opera principalmente en las siguientes tablas:

* publicacion: Contiene información detallada sobre las publicaciones de libros en el sistema, incluidos detalles específicos como el stock y los precios de los libros.
* ficha\_libro: Almacena datos exhaustivos sobre los libros disponibles en el sistema, como el género al que pertenece cada libro.

Función 2: PRECIO\_PROMEDIO

Descripción:

Esta funcion se utiliza para calcular el precio promedio de un libro específico, tomando todas las publicaciones que utilizan el mismo libro en cuestión . Esta función combina datos clave de la tabla publicacion para determinar el precio promedio de un libro particular basado en su ID.

Objetivo:

El objetivo principal de esta funcion es proporcionar una manera eficiente de calcular el precio promedio de un libro específico. Al realizar este cálculo, la función permite a los usuarios obtener rápidamente una idea clara del precio medio al que se vende un libro en particular.

Tablas/Datos:

Esta funcion se compone de los siguientes elementos clave:

* Parámetro de entrada: libro\_id - Un entero que representa el ID del libro para el cual se desea calcular el precio promedio.
* Variable local: prom - Un valor de punto flotante que almacena el precio promedio del libro especificado.
* Consulta SQL: La función utiliza una consulta SQL que calcula el promedio de precios de las publicaciones asociadas con el libro específico basado en el parámetro de entrada libro\_id.

Esta funcion opera principalmente en la siguiente tabla:

* publicacion: Contiene información detallada sobre las publicaciones de libros en el sistema, incluidos detalles específicos como el precio y el stock de los libros.

1. Stored Procedures

Procedure 1: SP\_ORDENAR\_TABLA

Descripción

Este stored procedure se utiliza para ordenar dinámicamente una tabla específica de acuerdo con un campo específico. Este procedimiento almacenado acepta dos parámetros: el nombre de la tabla que se va a ordenar y el nombre del campo según el cual se realizará la ordenación.:

Objetivo:

El propósito principal de este stored procedure es proporcionar una forma dinámica de ordenar cualquier tabla en la base de datos según un campo específico, lo que facilita la organización y el análisis de los datos de la tabla en cuestión.

Tablas/Datos:

Este stored procedure se compone de los siguientes elementos clave:

* Parámetros de entrada:

nombre\_tabla: Un parámetro de tipo VARCHAR que representa el nombre de la tabla que se va a ordenar.

orden\_campo: Un parámetro de tipo VARCHAR que representa el campo según el cual se ordenará la tabla.

* Variable local: @q - Una variable que almacena la consulta SQL dinámica para ordenar la tabla según el campo especificado.

Procedure 2: SP\_ELIMINAR\_ENVIO\_HASTA\_FECHA

Descripción:

Este stored procedure se utiliza para eliminar todos los registros de la tabla envio que sean anteriores o iguales a la fecha ingresada. Este stored procedure acepta un parámetro de tipo DATETIME que representa la fecha límite hasta la cual se eliminarán los registros de envío antiguos.

Objetivo:

El proposito principal de este stored procedure es proporcionar una forma rápida y eficiente de eliminar todos los registros de envío antiguos de la tabla envio en la base de datos hasta una fecha específica proporcionada como parámetro.

Tablas/Datos:

Este stored procedure se compone de los siguientes elementos clave:

* Parámetro de entrada: fecha - Un parámetro de tipo DATETIME que representa la fecha límite hasta la cual se eliminarán los registros de envío antiguos.
* Estructura condicional IF: El stored procedure utiliza una estructura condicional para verificar si la fecha proporcionada es anterior o igual a la fecha actual. Si esta condición se cumple, se ejecuta la operación de eliminación (esto se utiliza para proporcionar una capa de seguridad y evitar que se eliminen envíos futuros que serán los envíos pendientes que todavía no se hayan concretado).
* Comando DELETE: El stored procedure utiliza un comando DELETE para eliminar todos los registros de la tabla envio que tengan una fecha de envío anterior o igual a la fecha proporcionada.

1. Archivos SQL

4.1 Script creación de objetos:

<https://github.com/feliRios/SQL-Coderhouse-DesafioScriptInsercionDatos-Lopez-Rios>

El script de creación de objetos es el archivo **‘database\_creation.sql’** del link.

4.2 Script inserción de datos:

<https://github.com/feliRios/SQL-Coderhouse-DesafioScriptInsercionDatos-Lopez-Rios>

La inserción de datos se realiza con el archivo **‘MOCK\_DATA\_(registros\_insercion).csv’** y el asistente de importación de MySQL Workbench (paso a paso explicado en archivo **‘explicacion\_script\_insercion.pdf’** del link)

Nota: a posteriori también se debe ejecutar el archivo **‘script.sql’** del link <https://github.com/feliRios/SQL-Coderhouse-DesafioScriptVistas-Lopez-Rios> para la correcta visualizacion del funcionamiento de las vistas.

4.3 Script creación de Vistas:

<https://github.com/feliRios/SQL-Coderhouse-DesafioScriptVistas-Lopez-Rios>

El script de creación de vistas es el archivo **‘script.sql’** del link.

4.4 Script creación de Funciones:

<https://github.com/feliRios/SQL-Coderhouse-DesafioScriptFunciones-Lopez-Rios>

El script de creación de funciones es el archivo **‘funciones.sql’** del link.

4.5 Script creación de Stored Procedures:

<https://github.com/feliRios/SQL-Coderhouse-DesafioScriptSPs-Lopez-Rios>

El script de creación de stored procedures es el archivo **‘stored\_procedures.sql’** del link.

4.6 Script creación de Triggers:

<https://github.com/feliRios/SQL-Coderhouse-DesafioTriggers-Lopez-Rios>

El script de creación de triggers es el archivo **‘stored\_procedures.sql’** del link.