Sprint 2

Agregar funcionalidades de gestión de operaciones y esenciales a la clase Categoria para organizar los productos de la tienda de cómics.
Se requiere que se puedan filtrar los productos mediante las Categorías: Marvel y DC.
También que puedan enlistar todos los productos ordenados según su categoría alfabéticamente.

Si bien la Clase ya esta creada ahora debe funcionar con la base de datos, para ello debe poder realizar consultas a la BDD y poder traer la información de la misma para que se pueda trabajar en Python.

- Lo que se espera es que se desarrolle un programa dentro de una función llamada def_prog_category
- Dicho programa permitirá al usuario realizar la búsqueda de productos según las categorías disponibles. Se puede ocupar un menú y validar las opciones que ingrese por teclado.
- Debe mostrar los resultados en formato tabla (puede usar tabulate).
- Por lo pronto estara destinado solo para consulta, pero si se logra terminar con exito se podrá avanzar para realizar la modificación de los datos en la base de datos.

Agregar funcionalidades de gestión de operaciones y esenciales a la clase Categoria para organizar los productos de la tienda de cómics.
Se requiere que se puedan filtrar los productos mediante las Categorías: Marvel y DC.
También que puedan enlistar todos los productos ordenados según su categoría alfabéticamente.

- def_prog_category.py : Para agregar las funcionalidades requeridas a la clase `Categoria` y hacer que funcione con la base de datos, primero debes modificar la clase `Categoria` para que pueda realizar consultas a la base de datos =

—----------

```
import mysql.connector

class Categoria:

  def __init__(self, host, user, password, database):

    self.conexion = mysql.connector.connect(

      host=host,

      user=user,

      password=password,
```

```
            database=database
        )

    def cerrar_conexion(self):
        if self.conexion.is_connected():
            self.conexion.close()

    def filtrar_productos_por_categoria(self, nombre_categoria):
        try:
            cursor = self.conexion.cursor(dictionary=True)
            consulta = """
                        SELECT p.id_producto, p.nombre, p.descripcion, p.precio, p.stock,
p.imagen
                FROM Producto p
                INNER JOIN Categoria c ON p.id_categoria = c.id_categoria
                WHERE c.nombre_categoria = %s
                ORDER BY p.nombre
            """
            cursor.execute(consulta, (nombre_categoria,))
            productos = cursor.fetchall()
            cursor.close()
```

```
            return productos
        except mysql.connector.Error as err:
            print(f"Error al filtrar productos por categoría: {err}")
            return []

    def listar_todos_los_productos_por_categoria(self):
        try:
            cursor = self.conexion.cursor(dictionary=True)
            consulta = """
                    SELECT c.nombre_categoria, p.id_producto, p.nombre, p.descripcion,
p.precio, p.stock, p.imagen
                FROM Categoria c
                LEFT JOIN Producto p ON c.id_categoria = p.id_categoria
                ORDER BY c.nombre_categoria, p.nombre
            """
            cursor.execute(consulta)
            productos = cursor.fetchall()
            cursor.close()
            return productos
        except mysql.connector.Error as err:
            print(f"Error al listar todos los productos por categoría: {err}")
```

```
        return []
```

---

y luego usar esta clase en la función `def_prog_category`. Aquí tienes una versión modificada de la clase `Categoria` que incluye las funcionalidades necesarias:

```
import mysql.connector

class Categoria:

    def __init__(self, host, user, password, database):

        self.conexion = mysql.connector.connect(

            host=host,

            user=user,

            password=password,

            database=database

        )

    def cerrar_conexion(self):

        if self.conexion.is_connected():

            self.conexion.close()

    def crear_categoria(self, nombre):
```

```
        try:
            cursor = self.conexion.cursor()
                consulta = "INSERT INTO categories (name) VALUES
(%s)"
            cursor.execute(consulta, (nombre,))
            self.conexion.commit()
            cursor.close()
            return True
        except mysql.connector.Error as err:
            print(f"Error al crear la categoría: {err}")
            return False
    def leer_categorias(self):
        try:
            cursor = self.conexion.cursor()
            consulta = "SELECT * FROM categories ORDER BY name"
            cursor.execute(consulta)
            categorias = cursor.fetchall()
            cursor.close()
            return categorias
```

```
        except mysql.connector.Error as err:
            print(f"Error al leer las categorías: {err}")
            return []
    def filtrar_productos_por_categoria(self, nombre_categoria):
        try:
            cursor = self.conexion.cursor(dictionary=True)
            consulta = """
                            SELECT p.id_products, p.name, p.price,
p.description
                FROM products p
                    INNER JOIN categories c ON p.id_categories =
c.id_categories
                WHERE c.name = %s
                ORDER BY p.name
            """
            cursor.execute(consulta, (nombre_categoria,))
            productos = cursor.fetchall()
            cursor.close()
            return productos
        except mysql.connector.Error as err:
```

```
                print(f"Error al filtrar productos por categoría:
{err}")
            return []
    def listar_todos_los_productos_por_categoria(self):
        try:
            cursor = self.conexion.cursor(dictionary=True)
            consulta = """
                    SELECT c.name AS categoria, p.name, p.price,
p.description
                FROM products p
                    INNER JOIN categories c ON p.id_categories =
c.id_categories
                ORDER BY c.name, p.name
            """
            cursor.execute(consulta)
            productos = cursor.fetchall()
            cursor.close()
            return productos
        except mysql.connector.Error as err:
```

```
                print(f"Error al listar todos los productos por categoría: {err}")
            return []
```

Luego, puedes usar esta clase en tu función `def_prog_category` para realizar consultas y mostrar los resultados en formato de tabla. Aquí tienes un ejemplo de cómo podría verse `def_prog_category`:

```
from tabulate import tabulate  # Asegúrate de tener instalado el paquete tabulate


def def_prog_category():
    host = "tu_host"
    user = "tu_usuario"
    password = "tu_contraseña"
    database = "planetSuperheroesDB"

    categoria = Categoria(host, user, password, database)

    while True:
```

```
        print("\nMenú de Categorías:")
        print("1. Filtrar Productos por Categoría (Marvel)")
        print("2. Filtrar Productos por Categoría (DC)")
        print("3. Listar todos los Productos por Categoría")
        print("0. Volver al Menú Principal")

        opcion = input("Seleccione una opción: ")
        if opcion == "1":
                                        productos_marvel             =
categoria.filtrar_productos_por_categoria("Marvel")
            if productos_marvel:
                print(tabulate(productos_marvel, headers="keys",
tablefmt="fancy_grid"))
            else:
                        print("No se encontraron productos en la
categoría Marvel.")

        elif opcion == "2":
                                                    productos_dc    =
categoria.filtrar_productos_por_categoria("DC")
```

```
            if productos_dc:
                    print(tabulate(productos_dc, headers="keys",
tablefmt="fancy_grid"))
            else:
                    print("No se encontraron productos en la
categoría DC.")


        elif opcion == "3":
                                                productos_todos    =
categoria.listar_todos_los_productos_por_categoria()
            if productos_todos:
                  print(tabulate(productos_todos, headers="keys",
tablefmt="fancy_grid"))
            else:
                   print("No se encontraron productos en ninguna
categoría.")


        elif opcion == "0":
            categoria.cerrar_conexion()
            break
```

```
        else:
            print("Opción no válida. Intente de nuevo.")
if __name__ == "__main__":
    def_prog_category()
```

Este código primero crea una instancia de la clase `Categoria` con la información de la base de datos y luego utiliza los métodos de esta clase para realizar consultas y mostrar los resultados en formato de tabla utilizando la biblioteca `tabulate`. Asegúrese de instalar `tabulate` si aún no lo ha hecho usando `pip install tabulate`.