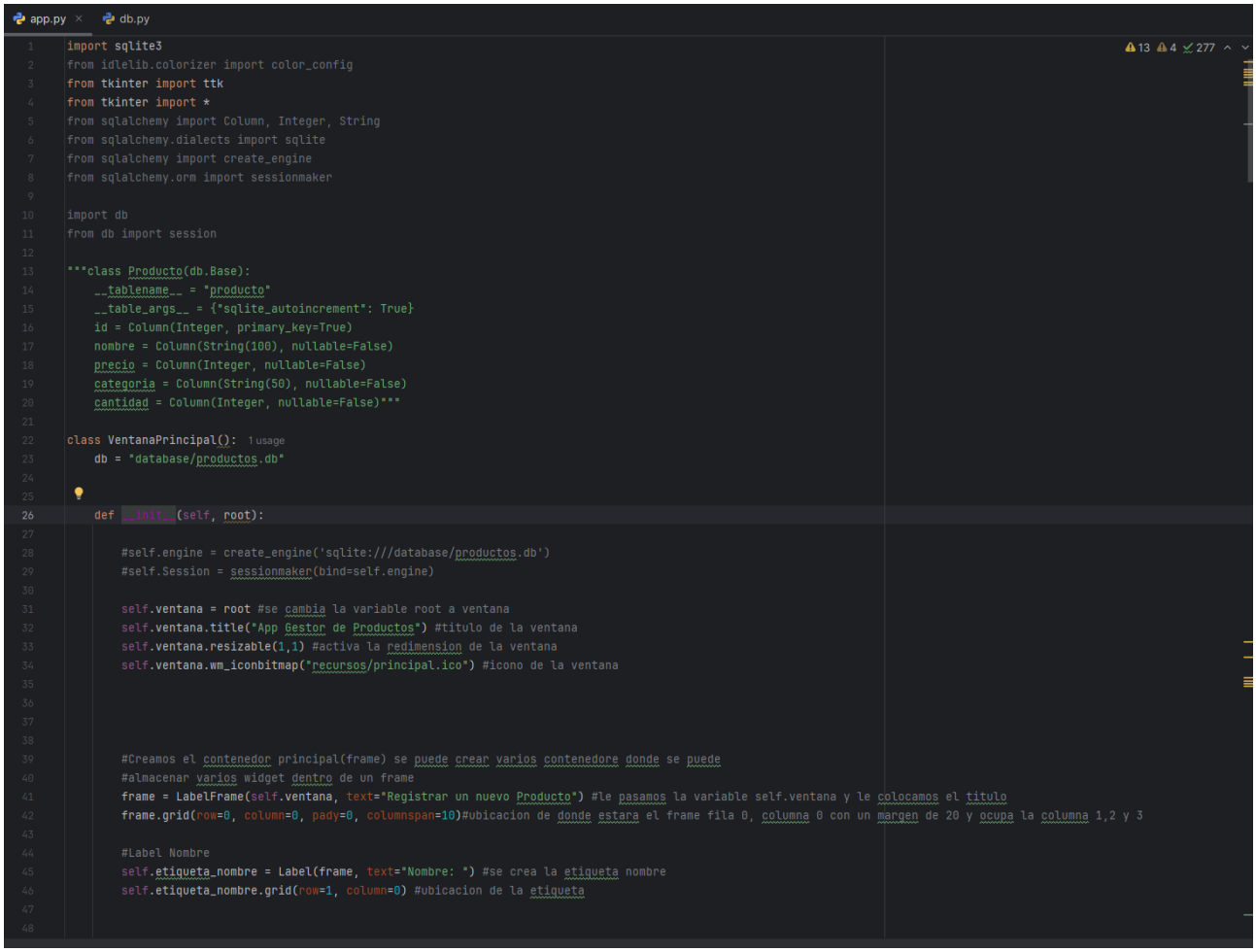
Entrega tarea M6-03

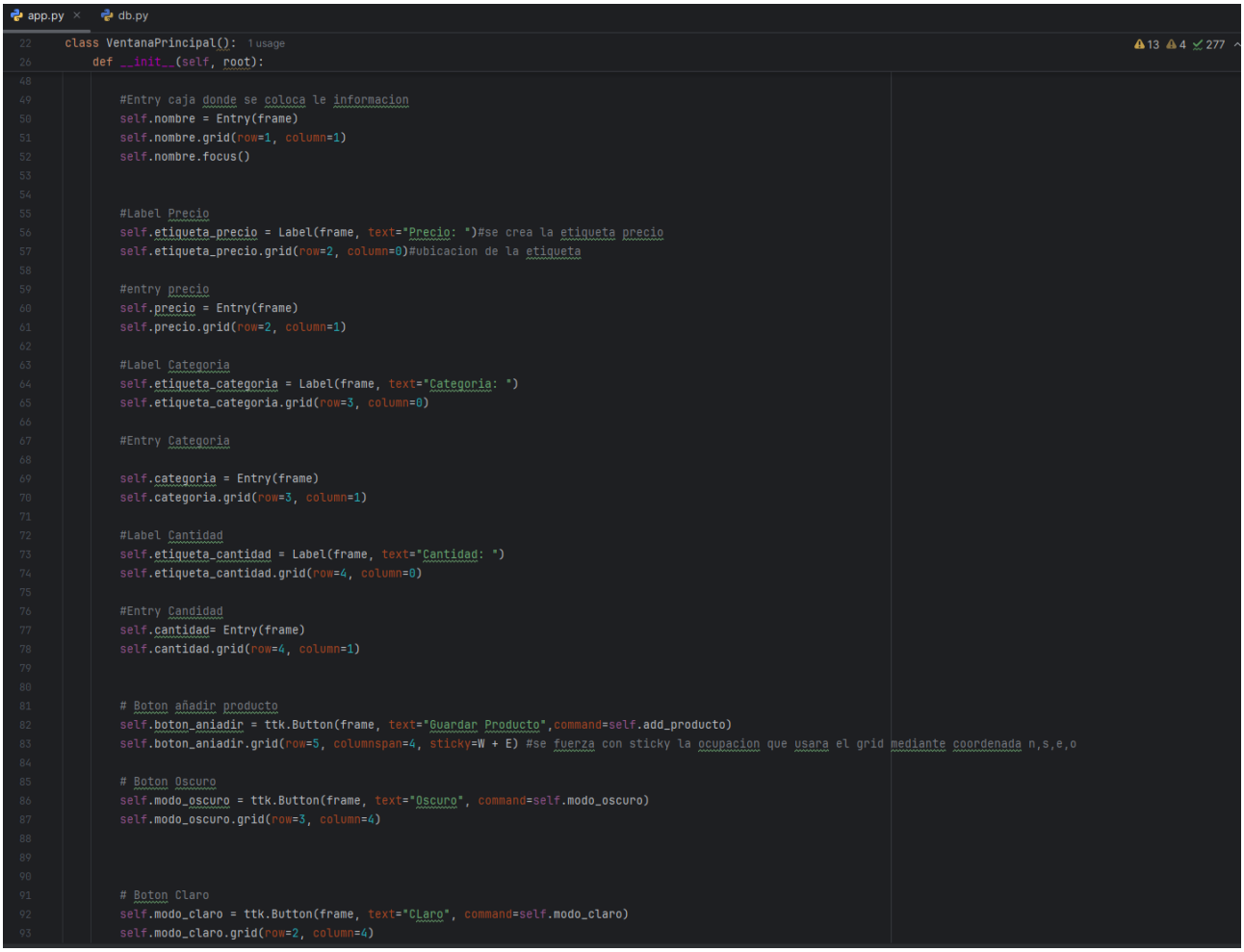
Buenas noches, primeramente saludarlo por ser un excelente profesor he aprendido mucho desde que inicie el curso y también he tenido varios dolores de cabeza pero es parte del proceso de aprendizaje, bueno intente en reemplazar el código sqlite3 al orm, pero varios días intentando, leyendo, investigando no pude llegar al cambio, asi que bueno en las próximas capturas de pantallas, veras varias líneas de código comentadas por el intento fallido, asi que le pido que me explique como seria reemplazar el código al orm.

En las próximas capturas hice varios cambios:

1. intente remplazar el código sqlite2 al orm, por eso hay varios import en los inicios.

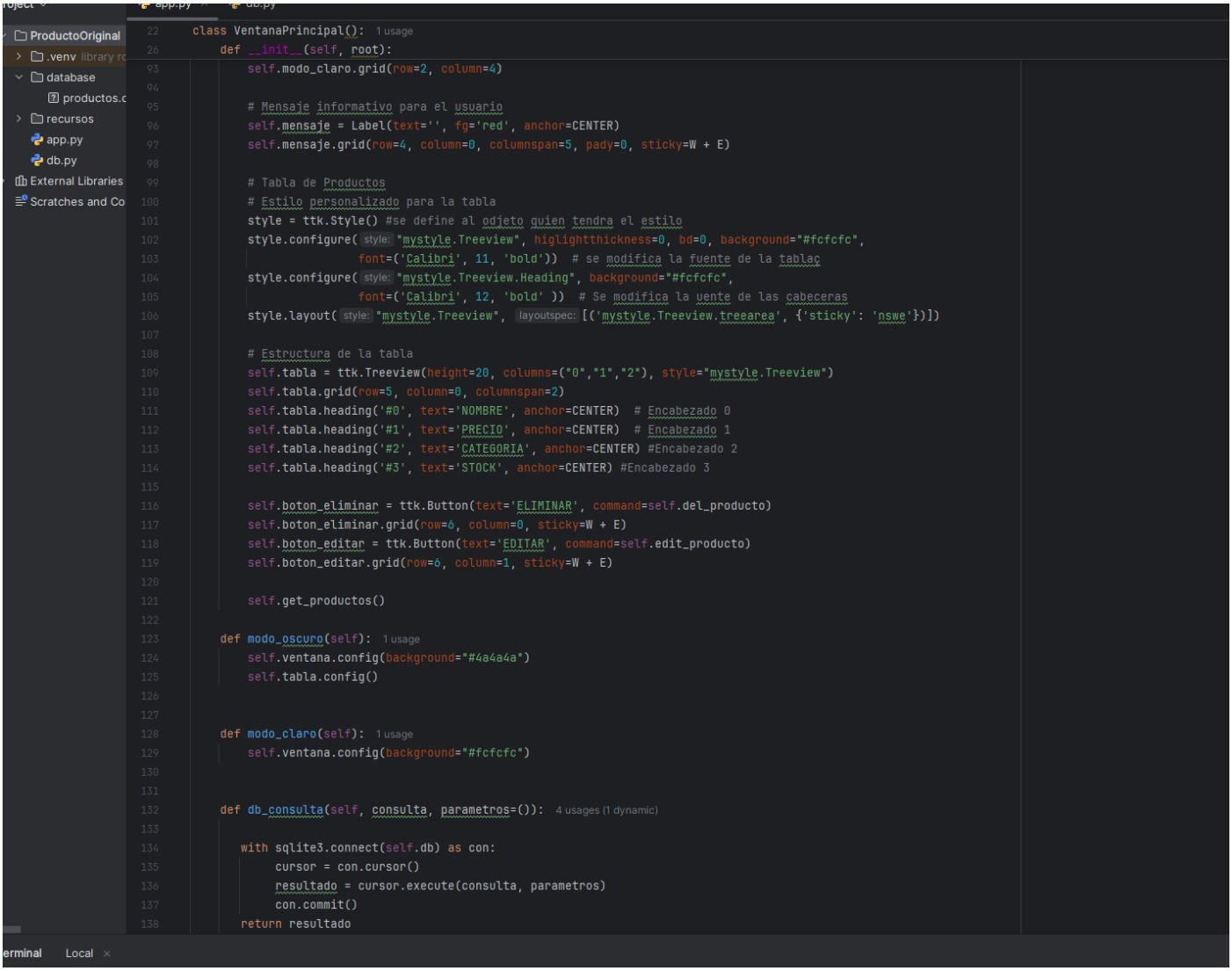


2. Creo el Label/Entry de categoría y cantidad asi se muestra en la ventana principal, también cree dos funciones que era cambiar el tema de la app, pero no logre hacerlo en la ventana completa, asi que vera esa función a media.

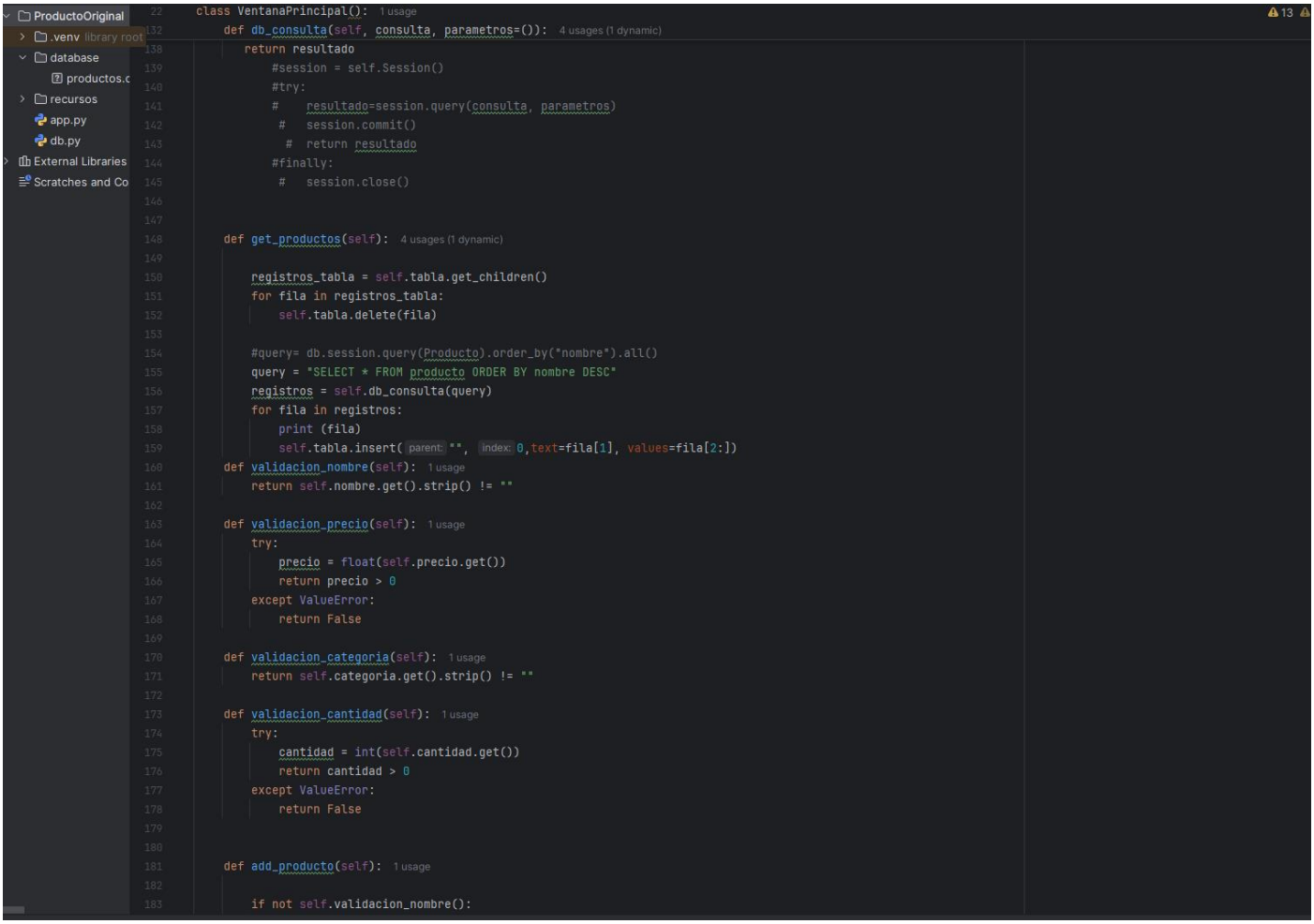


```python
app.py  db.py

class VentanaPrincipal():  1 usage
    def __init__(self, root):

        #Entry caja donde se coloca le informacion
        self.nombre = Entry(frame)
        self.nombre.grid(row=1, column=1)
        self.nombre.focus()



        #Label Precio
        self.etiqueta_precio = Label(frame, text="Precio: ")#se crea la etiqueta precio
        self.etiqueta_precio.grid(row=2, column=0)#ubicacion de la etiqueta

        #entry precio
        self.precio = Entry(frame)
        self.precio.grid(row=2, column=1)

        #Label Categoria
        self.etiqueta_categoria = Label(frame, text="Categoria: ")
        self.etiqueta_categoria.grid(row=3, column=0)

        #Entry Categoria

        self.categoria = Entry(frame)
        self.categoria.grid(row=3, column=1)

        #Label Cantidad
        self.etiqueta_cantidad = Label(frame, text="Cantidad: ")
        self.etiqueta_cantidad.grid(row=4, column=0)

        #Entry Candidad
        self.cantidad= Entry(frame)
        self.cantidad.grid(row=4, column=1)



        # Boton añadir producto
        self.boton_aniadir = ttk.Button(frame, text="Guardar Producto",command=self.add_producto)
        self.boton_aniadir.grid(row=5, columnspan=4, sticky=W + E) #se fuerza con sticky la ocupacion que usara el grid mediante coordenada n,s,e,o

        # Boton Oscuro
        self.modo_oscuro = ttk.Button(frame, text="Oscuro", command=self.modo_oscuro)
        self.modo_oscuro.grid(row=3, column=4)




        # Boton Claro
        self.modo_claro = ttk.Button(frame, text="CLaro", command=self.modo_claro)
        self.modo_claro.grid(row=2, column=4)
```

3. En la tabla que tiene su estilo, le modifique la cantidad de columna que tendrá ya que se creo varios label como la categoría/cantidad. También esta el método modo_oscuro/claro, como le mencione que no pude cambiarle el color de fondo en todo el programa hice varios intentos, modificaciones sin lograrlo.



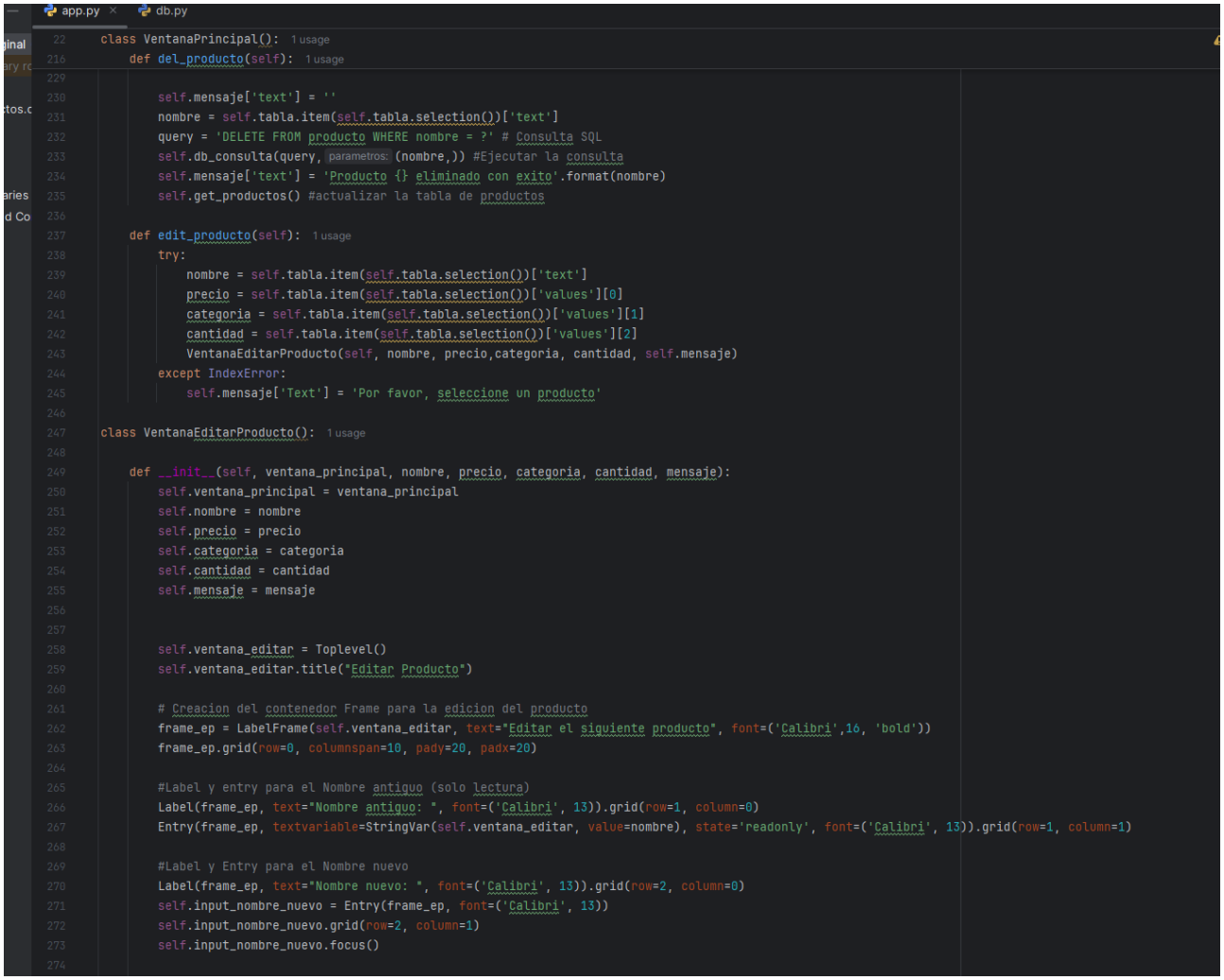
4. En el método get_productos intente conectarme a sqlalchemy, pero me dio varios errores, modifique el código muchísimas veces sin lograrlo.



5. En este punto hice varios cambios, me costo pero lo logre:
    a. En el query, agregue los signos de interrogación faltante.
    b. Luego en los parámetros se agrego las variables nuevas, categoría, cantidad

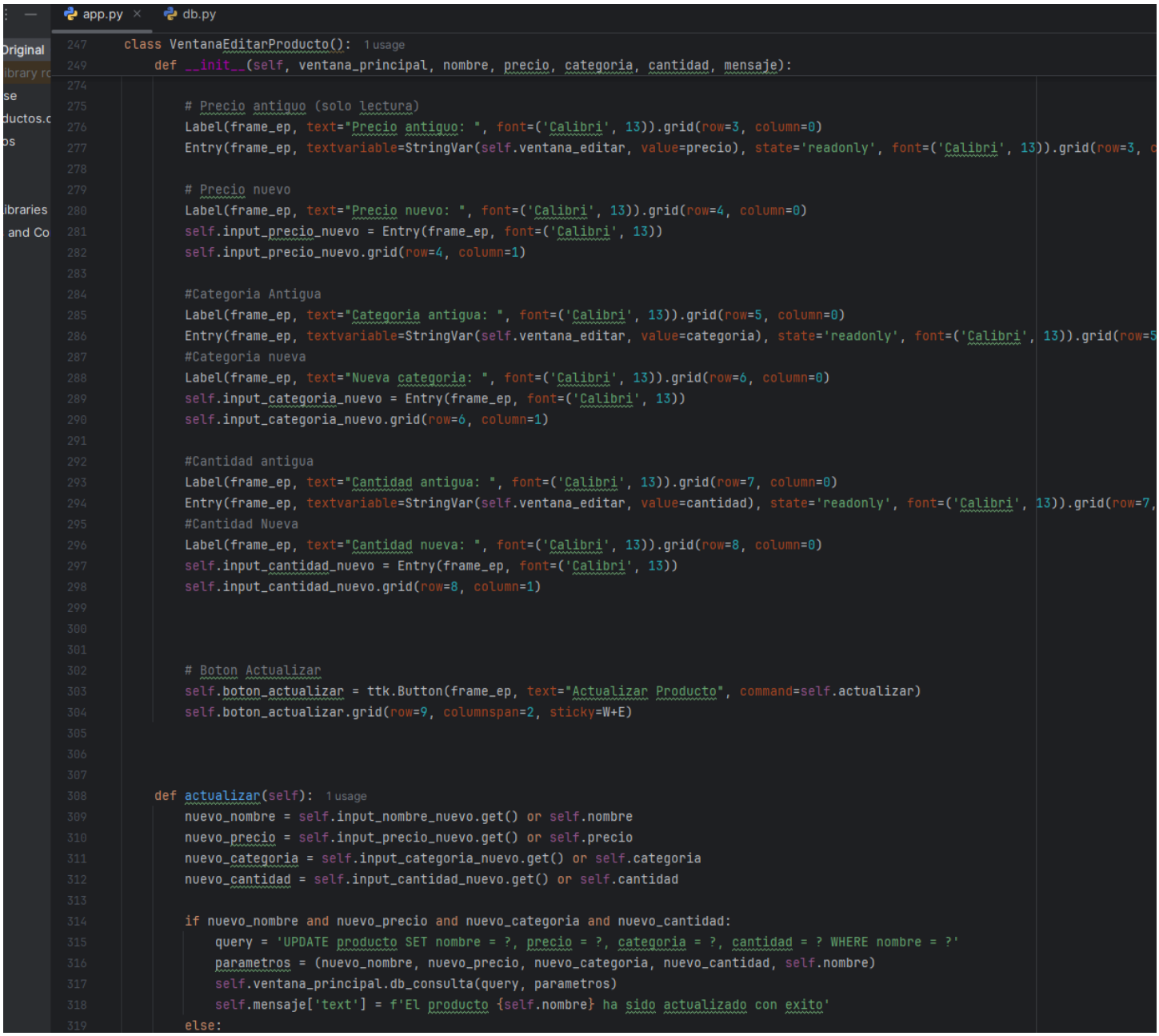
```
22    class VentanaPrincipal():  1 usage
181       def add_producto(self):  1 usage
183           if not self.validacion_nombre():
184               print("El nombre es obligatorio")
185               self.mensaje['text'] = 'El nombre es oligatorio y no puede estar vacio'
186               return
187           if not self.validacion_precio():
188               print("El Precio es obligatorio")
189               self.mensaje['text'] = 'El precio es obligatorio y debe ser un numero valido mayor que 0'
190               return
191           if not self.validacion_categoria():
192               print("La categoria es obligatorio")
193               self.mensaje['text'] = 'La Categoria es oligatorio y no puede estar vacio'
194               return
195           if not self.validacion_cantidad():
196               print("La cantidad es obligatorio")
197               self.mensaje['text'] = 'La cantidad es obligatorio y debe ser un numero valido mayor que 0'
198               return
199
200
201
202           query = 'INSERT INTO producto VALUES(NULL, ?, ?,?,?)'
203           parametros = (self.nombre.get(), self.precio.get(), self.categoria.get(), self.cantidad.get())
204           self.db_consulta(query, parametros)
205           print("Datos Guardados")
206           self.mensaje['text'] = 'producto {} añadido con exito'.format(self.nombre.get())
207           self.nombre.delete( first: 0, END) #Borrar el campo nombre del formulario
208           self.precio.delete( first: 0, END) #Borrar el campo precio del formulario
209           self.categoria.delete( first: 0, END)
210           self.cantidad.delete( first: 0, END)
211           #print(self.nombre.get())
212           #print(self.precio.get())
213
214           self.get_productos()
215
216       def del_producto(self):  1 usage
217           # print(self.tabla.item(self.tabla.selection()))
218           #print(self.tabla.item(self.tabla.selection())['text'])
219           # print(self.tabla.item(self.tabla.selection())['values'])
220           #print(self.tabla.item(self.tabla.selection())['values'][0])
221
222           self.mensaje['text'] = '' # Mensaje inicialmente vacio
223           # Comprobacion de que se seleccione un procuto para poder eliminarlo
224           try:
225               self.tabla.item(self.tabla.selection())['text'][0]
226           except IndexError as e:
227               self.mensaje['text'] = 'Por favor, seleccione un producto'
228               return
```

c. En el método edit_producto tiene las variables categoría, cantidad.
d. En el método VentanaEditarProducto también se agrego las nuevas variables



```python
class VentanaPrincipal():  1 usage
    def del_producto(self):  1 usage

        self.mensaje['text'] = ''
        nombre = self.tabla.item(self.tabla.selection())['text']
        query = 'DELETE FROM producto WHERE nombre = ?' # Consulta SQL
        self.db_consulta(query, parametros: (nombre,)) #Ejecutar la consulta
        self.mensaje['text'] = 'Producto {} eliminado con exito'.format(nombre)
        self.get_productos() #actualizar la tabla de productos

    def edit_producto(self):  1 usage
        try:
            nombre = self.tabla.item(self.tabla.selection())['text']
            precio = self.tabla.item(self.tabla.selection())['values'][0]
            categoria = self.tabla.item(self.tabla.selection())['values'][1]
            cantidad = self.tabla.item(self.tabla.selection())['values'][2]
            VentanaEditarProducto(self, nombre, precio,categoria, cantidad, self.mensaje)
        except IndexError:
            self.mensaje['Text'] = 'Por favor, seleccione un producto'

class VentanaEditarProducto():  1 usage

    def __init__(self, ventana_principal, nombre, precio, categoria, cantidad, mensaje):
        self.ventana_principal = ventana_principal
        self.nombre = nombre
        self.precio = precio
        self.categoria = categoria
        self.cantidad = cantidad
        self.mensaje = mensaje


        self.ventana_editar = Toplevel()
        self.ventana_editar.title("Editar Producto")

        # Creacion del contenedor Frame para la edicion del producto
        frame_ep = LabelFrame(self.ventana_editar, text="Editar el siguiente producto", font=('Calibri',16, 'bold'))
        frame_ep.grid(row=0, columnspan=10, pady=20, padx=20)

        #Label y entry para el Nombre antiguo (solo lectura)
        Label(frame_ep, text="Nombre antiguo: ", font=('Calibri', 13)).grid(row=1, column=0)
        Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=nombre), state='readonly', font=('Calibri', 13)).grid(row=1, column=1)

        #Label y Entry para el Nombre nuevo
        Label(frame_ep, text="Nombre nuevo: ", font=('Calibri', 13)).grid(row=2, column=0)
        self.input_nombre_nuevo = Entry(frame_ep, font=('Calibri', 13))
        self.input_nombre_nuevo.grid(row=2, column=1)
        self.input_nombre_nuevo.focus()
```

e. la ventana editar también se agregaron los nuevos cambios de las variables nuevas me costo un monto manejar las filas/columnas de la ventana.
f. En el query UPDATE también se agregaron las variables nuevas



```
app.py  db.py

class VentanaEditarProducto():  1 usage
    def __init__(self, ventana_principal, nombre, precio, categoria, cantidad, mensaje):

        # Precio antiguo (solo lectura)
        Label(frame_ep, text="Precio antiguo: ", font=('Calibri', 13)).grid(row=3, column=0)
        Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=precio), state='readonly', font=('Calibri', 13)).grid(row=3, c

        # Precio nuevo
        Label(frame_ep, text="Precio nuevo: ", font=('Calibri', 13)).grid(row=4, column=0)
        self.input_precio_nuevo = Entry(frame_ep, font=('Calibri', 13))
        self.input_precio_nuevo.grid(row=4, column=1)

        #Categoria Antigua
        Label(frame_ep, text="Categoria antigua: ", font=('Calibri', 13)).grid(row=5, column=0)
        Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=categoria), state='readonly', font=('Calibri', 13)).grid(row=5
        #Categoria nueva
        Label(frame_ep, text="Nueva categoria: ", font=('Calibri', 13)).grid(row=6, column=0)
        self.input_categoria_nuevo = Entry(frame_ep, font=('Calibri', 13))
        self.input_categoria_nuevo.grid(row=6, column=1)

        #Cantidad antigua
        Label(frame_ep, text="Cantidad antigua: ", font=('Calibri', 13)).grid(row=7, column=0)
        Entry(frame_ep, textvariable=StringVar(self.ventana_editar, value=cantidad), state='readonly', font=('Calibri', 13)).grid(row=7,
        #Cantidad Nueva
        Label(frame_ep, text="Cantidad nueva: ", font=('Calibri', 13)).grid(row=8, column=0)
        self.input_cantidad_nuevo = Entry(frame_ep, font=('Calibri', 13))
        self.input_cantidad_nuevo.grid(row=8, column=1)



        # Boton Actualizar
        self.boton_actualizar = ttk.Button(frame_ep, text="Actualizar Producto", command=self.actualizar)
        self.boton_actualizar.grid(row=9, columnspan=2, sticky=W+E)



    def actualizar(self):  1 usage
        nuevo_nombre = self.input_nombre_nuevo.get() or self.nombre
        nuevo_precio = self.input_precio_nuevo.get() or self.precio
        nuevo_categoria = self.input_categoria_nuevo.get() or self.categoria
        nuevo_cantidad = self.input_cantidad_nuevo.get() or self.cantidad

        if nuevo_nombre and nuevo_precio and nuevo_categoria and nuevo_cantidad:
            query = 'UPDATE producto SET nombre = ?, precio = ?, categoria = ?, cantidad = ? WHERE nombre = ?'
            parametros = (nuevo_nombre, nuevo_precio, nuevo_categoria, nuevo_cantidad, self.nombre)
            self.ventana_principal.db_consulta(query, parametros)
            self.mensaje['text'] = f'El producto {self.nombre} ha sido actualizado con exito'
        else:
```

Bueno este es el fin de la app, se que los cambios no parecen mucho pero si me costo un montón mas en el intento de reemplazo de la orm, me gustaría que me indicara cual es la solución. Asi que bueno. Muchas gracias.