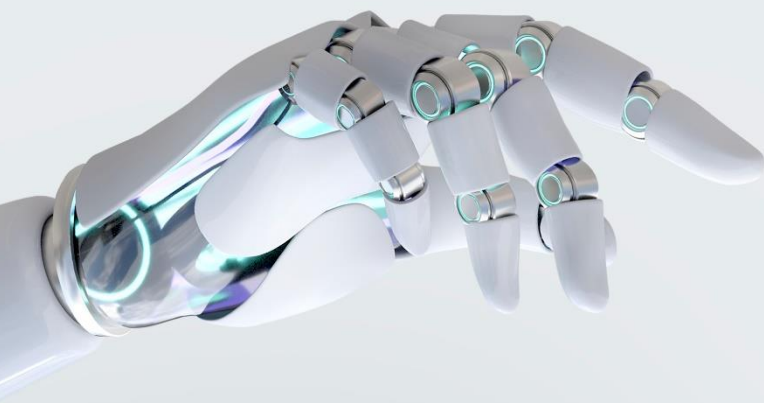


Gestión Empresarial

20-11-2024

SGE_H2_1T_Miguel_Gonzalez_Galan



Autor: Miguel González Galán

ÍNDICE

1.Introducción	3
2.Interfaz de Usuario y Conexión de Base de Datos	3
2.1.Explicación de la conexión MySQL.....	3
2.2.Descripción de la interfaz Tkinter	3
2.2.1.Estructura Principal	3
2.2.2.Menú de Aplicación.....	4
2.2.3.Botones Principales	4
2.2.4.Tabla de Datos (Treeview)	4
2.2.5.Formulario de Edición/Añadir	5
2.3.Interfaz visual	5
3.Operaciones CRUD	6
4.Consultas y Ordenación	7
5.Visualización de Datos.....	9
6.Código Comentado	10
7.Referencias bibliográficas	11

1. Introducción

Contexto y Objetivo El consumo de alcohol es un problema de salud pública complejo que requiere herramientas de análisis precisas y eficientes. Esta aplicación surge de la necesidad de:

- Recopilar datos sistemáticos sobre hábitos de consumo
- Facilitar el análisis estadístico de patrones de consumo
- Relacionar el consumo de alcohol con indicadores de salud

Problema a Resolver En la actualidad, la recopilación y análisis de datos sobre consumo de alcohol presenta varios desafíos:

- Dispersión y fragmentación de la información
- Dificultad para realizar análisis comparativos
- Necesidad de herramientas que faciliten la visualización de datos

```
campos_encuesta = [  
    "Edad", "Sexo", "BebidasSemana", "CervezasSemana",  
    "BebidasFinSemana", "BebidasDestiladasSemana",  
    "VinosSemana", "PerdidasControl",  
    "DiversiónDependenciaAlcohol",  
    "ProblemasDigestivos", "TensionAlta", "DolorCabeza"  
]
```

2. Interfaz de Usuario y Conexión de Base de Datos

2.1. Explicación de la conexión MySQL

```
def create_connection():  
    connection = pymysql.connect(  
        host="localhost",  
        user="root",  
        password="curso",  
        database="ENCUESTAS",  
        cursorclass=pymysql.cursors.DictCursor  
    )  
    return connection
```

2.2. Descripción de la interfaz Tkinter

2.2.1. Estructura Principal

```
def __init__(self, root):  
    self.root = root  
    self.root.title("Gestión de Encuestas")  
    self.root.configure(bg="#f0f0f0") # Color de fondo gris claro  
    self.create_widgets()
```

2.2.2. Menú de Aplicación

```
def create_widgets(self):
    # Barra de menú
    menubar = tk.Menu(self.root)
    file_menu = tk.Menu(menubar, tearoff=0)
    menubar.add_cascade(label="Archivo", menu=file_menu)
    file_menu.add_command(label="Salir", command=self.root.quit)
```

2.2.3. Botones Principales

```
# Estilo de botones consistente
button_style = {
    "bg": "#4CAF50", # Verde
    "fg": "white",    # Texto blanco
    "font": ("Arial", 12),
    "width": 15,
    "height": 2
}

# Creación de botones con funcionalidades específicas
tk.Button(frame, text="Añadir Encuesta", command=self.add_encuesta,
**button_style)
tk.Button(frame, text="Actualizar Encuesta", command=self.update_encuesta,
**button_style)
tk.Button(frame, text="Eliminar Encuesta", command=self.delete_encuesta,
**button_style)
tk.Button(frame, text="Filtrar Encuestas", command=self.filter_encuestas,
**button_style)
tk.Button(frame, text="Exportar a Excel", command=self.export_to_excel,
**button_style)
tk.Button(frame, text="Visualizar Gráficos",
command=self.show_visualization_menu, **button_style)
```

2.2.4. Tabla de Datos (Treeview)

```
def create_widgets(self):
    # Configuración de estilo para la tabla
    style = ttk.Style()
    style.configure("Treeview.Heading",
                    font=("Arial", 10, "bold"),
                    background="#4CAF50",
                    foreground="black")

    # Columnas de la tabla
    self.tree = ttk.Treeview(self.root, columns=(
        "idEncuesta", "edad", "Sexo",
        "BebidasSemana", "CervezasSemana",
        "BebidasFinSemana", "BebidasDestiladasSemana",
        "VinosSemana", "PerdidasControl",
        "DiversiónDependenciaAlcohol",
        "ProblemasDigestivos", "TensionAlta",
        "DolorCabeza"
```

```

), show="headings")

# Configurar cada columna
for col in self.tree["columns"]:
    self.tree.heading(col,
                       text=col,
                       command=lambda _col=col: self.sort_by(_col))
    self.tree.column(col, width=100, anchor='center')

```

2.2.5. Formulario de Edición/Añadir

```

def edit_encuesta(self, values=None):
    ventana = tk.Toplevel(self.root)
    ventana.title("Añadir/Editar Encuesta")

    # Etiquetas y campos del formulario
    labels = [
        "ID Encuesta", "Edad", "Sexo", "Bebidas Semana",
        "Cervezas Semana", "Bebidas Fin de Semana",
        "Bebidas Destiladas Semana", "Vinos Semana",
        "Pérdidas de Control", "Diversión Dependencia Alcohol",
        "Problemas Digestivos", "Tensión Alta", "Dolor de Cabeza"
    ]

    # Campos especiales con opciones predefinidas
    options = {
        "Sexo": ["Masculino", "Femenino", "Otro"],
        "Dolor de Cabeza": ["Muy a menudo", "Alguna vez", "Nunca"],
        "Tensión Alta": ["Sí", "No", "No lo sé"],
        "Problemas Digestivos": ["Sí", "No"],
        "Diversión Dependencia Alcohol": ["Sí", "No"]
    }

```

2.3. Interfaz visual

Gestión de Encuestas

archivo

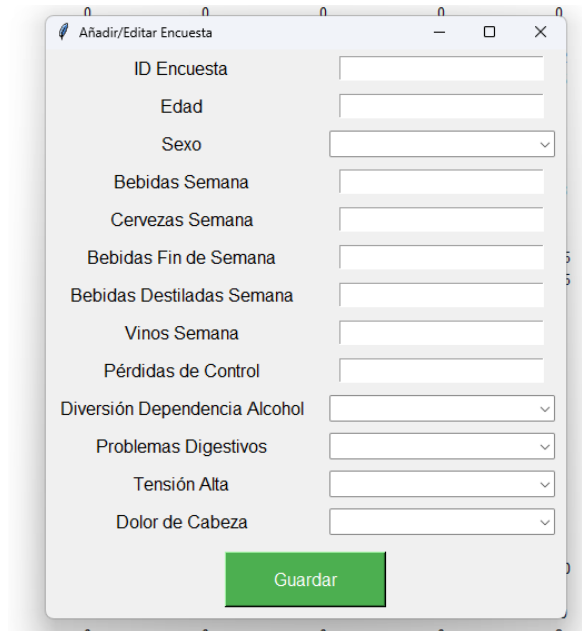
Añadir Encuesta	Actualizar Encuesta	Eliminar Encuesta	Filtrar Encuestas	Exportar a Excel	Visualizar Gráficos	Volver
-----------------	---------------------	-------------------	-------------------	------------------	---------------------	--------

idEncuesta	edad	Sexo	BebidasSemana	CervezasSemana	BebidasFinSem	BebidasDestiladas	VinosSemana	PerdidasControl	DiversiónDepen	ProblemasDige	TensionAlta	DolorCabeza
1	57	Mujer	5	5	5	0	3	1	No	Sí	No lo se	Alguna vez
2	55	Mujer	6	4	5	1	2	3	Sí	Sí	No	Alguna vez
3	19	Hombre	0	0	0	0	0	0	No	No	No lo se	Alguna vez
4	20	Hombre	0	0	0	0	0	0	No	No	No	Nunca
5	21	Hombre	20	15	10	5	0	12	Sí	No	No	Alguna vez
6	20	Hombre	1	12	45	12	23	45	No	No	No	Alguna vez
7	19	Hombre	2	0	2	2	0	3	No	No	No	Nunca
8	19	Hombre	33	5	5	5	0	1	Sí	No	No lo se	Muy a menudo
9	22	Hombre	0	0	0	0	0	0	No	No	No lo se	Alguna vez
10	19	Hombre	2	10	5	1	5	2	No	No	No	Alguna vez
11	19	Hombre	83	0	75	27	56	33	Sí	Sí	Sí	Muy a menudo
12	21	Hombre	0	0	0	0	0	0	No	No	No	Alguna vez
13	19	Hombre	5	5	5	5	0	1	Sí	No	No lo se	Alguna vez
14	24	Hombre	25	18	10	7	0	365	No	No	No	Alguna vez
15	38	Mujer	41	5	25	10	10	365	Sí	Sí	Sí	Muy a menudo
16	53	Mujer	0	0	0	0	0	0	No	No	Sí	Alguna vez
17	19	Hombre	4	0	1	1	1	1	No	No	No	Nunca

3. Operaciones CRUD

- Crear Encuesta:

```
def create_encuesta(connection, datos):  
    query = """  
    INSERT INTO ENCUESTA (idEncuesta, edad, Sexo, ...)  
    VALUES (%s, %s, %s, ...)  
    """  
    cursor.execute(query, datos)
```



- Leer Encuestas:

```
def read_encuestas(connection, order_by=None, filter_by=None):  
    query = "SELECT * FROM ENCUESTA"  
    if filter_by:  
        query += f" WHERE {filter_by}"  
    if order_by:  
        query += f" ORDER BY {order_by}"
```

- Actualizar Encuesta:

```
def update_encuesta(connection, datos):
    query = """
    UPDATE ENCUESTA SET edad=%s, Sexo=%s, ...
    WHERE idEncuesta=%s
    """
```

Añadir/Editar Encuesta

ID Encuesta	2
Edad	55
Sexo	Mujer
Bebidas Semana	6
Cervezas Semana	4
Bebidas Fin de Semana	5
Bebidas Destiladas Semana	1
Vinos Semana	2
Pérdidas de Control	3
Diversión Dependencia Alcohol	Sí
Problemas Digestivos	Sí
Tensión Alta	No
Dolor de Cabeza	Alguna vez

Guardar

- Eliminar Encuesta:

```
def delete_encuesta(connection, idEncuesta):
    query = "DELETE FROM ENCUESTA WHERE idEncuesta=%s"
```

4. Consultas y Ordenación

- Método de filtrado:

```
def filter_encuestas(self):
    filtro = f"{campo}='{valor}'"
    self.load_data(filter_by=filtro)
```

Filtrar Encuestas

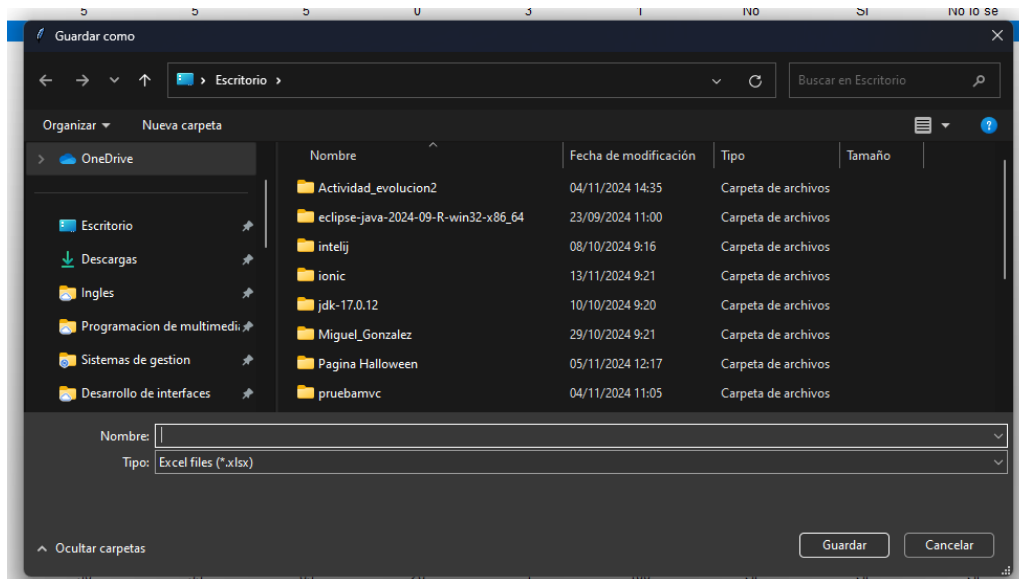
Campo:

Valor:

Aplicar Filtro

- Exportación a Excel

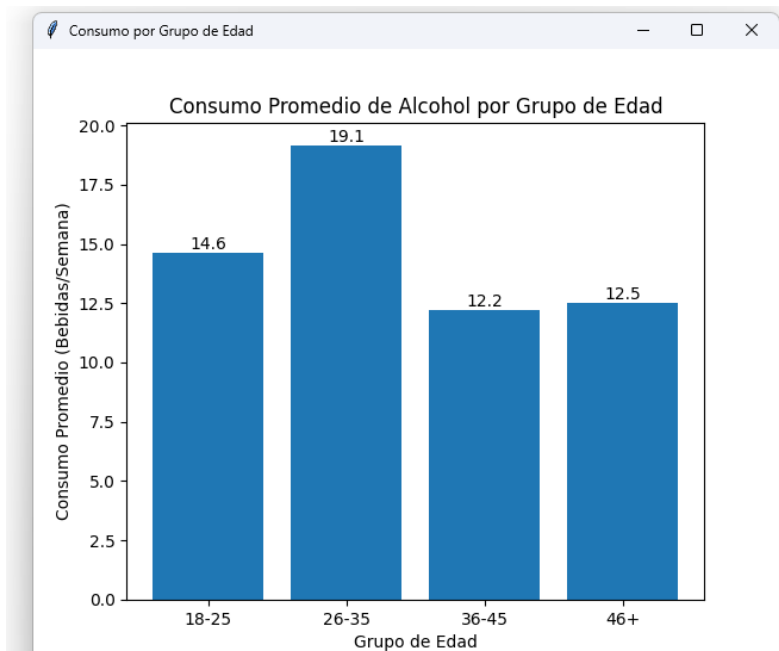
```
def export_to_excel(self):  
    data = []  
    for item in self.tree.get_children():  
        data.append(self.tree.item(item)["values"])  
  
    df = pd.DataFrame(data, columns=[...])  
    df.to_excel(file_path, index=False)
```



5. Visualización de Datos

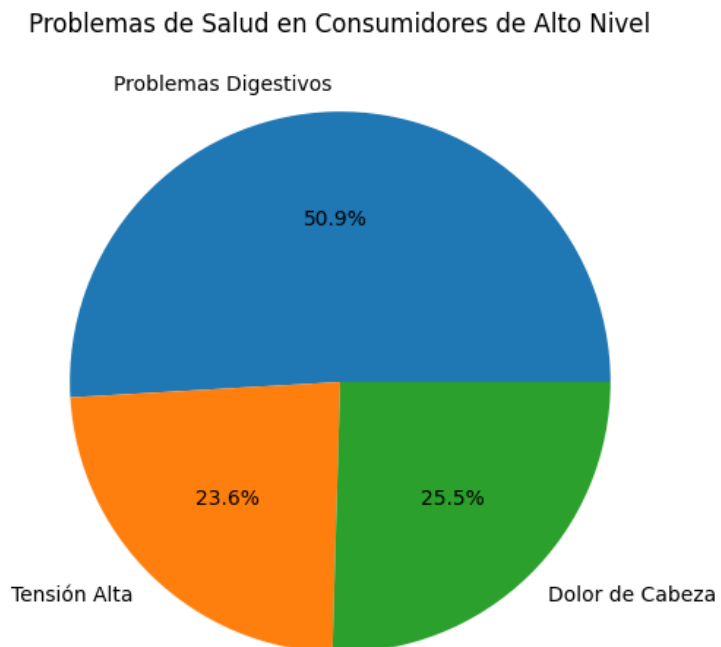
- Gráfico de Consumo por Grupo de Edad:

```
def show_age_consumption_graph(self, parent_window):  
    # Procesar datos por grupos de edad  
    age_groups = ['18-25', '26-35', '36-45', '46+']  
    averages = [np.mean(group_consumption[group]) for group in age_groups]  
  
    # Crear gráfico de barras  
    fig, ax = plt.subplots()  
    ax.bar(age_groups, averages)
```



- Gráfico de Correlación Alcohol y Salud:

```
def show_health_correlation_graph(self, parent_window):  
    # Calcular problemas de salud en consumidores con alto consumo  
    health_issues = {  
        'Problemas Digestivos': 0,  
        'Tensión Alta': 0,  
        'Dolor de Cabeza': 0  
    }  
  
    # Crear gráfico circular  
    fig, ax = plt.subplots()  
    ax.pie(sizes, labels=list(health_issues.keys()), autopct='%1.1f%%')
```



6. Código Comentado

- Enlace al repositorio de GitHub:
https://github.com/gmiguelg0/SGE_H2_1T_Miguel_Gonzalez_Galan.git

7. Referencias bibliográficas

1. *Java print/display variables*. (n.d.). W3schools.com. Retrieved November 18, 2024, from https://www.w3schools.com/java/java_variables_print.asp
2. *Java tutorial*. (n.d.). W3schools.com. Retrieved November 18, 2024, from <https://www.w3schools.com/java/default.asp>
3. *Matplotlib tutorial*. (n.d.). W3schools.com. Retrieved November 18, 2024, from https://www.w3schools.com/python/matplotlib_intro.asp

4. *MySQL tutorial*. (n.d.). W3schools.com. Retrieved November 18, 2024, from <https://www.w3schools.com/mysql/default.asp>
5. *Pandas tutorial*. (n.d.). W3schools.com. Retrieved November 18, 2024, from <https://www.w3schools.com/python/pandas/default.asp>