

Sistema de gestión empresarial

Segundo Hito – Primer Trimestre



20 de noviembre de 2024

andrea aparicio garcía

Campus FP

Contenido

[Introducción 3](#_Toc182556586)

[Descripción de la Interfaz y Conexión 3](#_Toc182556587)

[Etiquetas y botones 4](#_Toc182556588)

[Funciones ComboBox 5](#_Toc182556589)

[Operaciones CRUD 6](#_Toc182556590)

[Ingresar 6](#_Toc182556591)

[Lógica 6](#_Toc182556592)

[Código 6](#_Toc182556593)

[Mostrar 6](#_Toc182556594)

[Lógica 6](#_Toc182556595)

[Código 7](#_Toc182556596)

[Actualizar 7](#_Toc182556597)

[Lógica 7](#_Toc182556598)

[Código 7](#_Toc182556599)

[Eliminar 7](#_Toc182556600)

[Lógica 8](#_Toc182556601)

[Código 8](#_Toc182556602)

[Consultas y Ordenación de Datos 8](#_Toc182556603)

[Seleccionar Persona 8](#_Toc182556604)

[Lógica 8](#_Toc182556605)

[Código 9](#_Toc182556606)

[Limpiar Campos 9](#_Toc182556607)

[Lógica 9](#_Toc182556608)

[Código 9](#_Toc182556609)

[Filtrar Datos 10](#_Toc182556610)

[Lógica 10](#_Toc182556611)

[Código 10](#_Toc182556612)

[Exportación de Resultados a Excel 11](#_Toc182556613)

[Lógica 11](#_Toc182556614)

[Código 12](#_Toc182556615)

[Visualización de Datos en Gráficos 12](#_Toc182556616)

[Gráfico de Barras 12](#_Toc182556617)

[Personalización 12](#_Toc182556618)

[Código comentado 14](#_Toc182556619)

# Introducción

El **consumo de alcohol** es un factor que afecta directamente en la salud pública. Si se consume en exceso, puede provocar una variedad de problemas, desde trastornos de salud física (como problemas digestivos, dolores de cabeza, o tensión alta) hasta problemas de salud mental y conductas de riesgo (como la pérdida de control). En este contexto, es importante entender el patrón de consumo de alcohol de distintos segmentos de la población y cómo esto se relaciona con síntomas o condiciones de salud.

Con la base de datos ‘*encuesta’* hemos podido **analizar las tendencias y patrones** que podrían asociarse con diferentes niveles y tipos de consumo de alcohol (cerveza, vino, bebidas destiladas, etc.) y determinar si existe una posible relación con problemas de salud específicos reportados por los encuestados.

Descripción de la Interfaz y Conexión

Para este proyecto, he diseñado una interfaz de usuario utilizando la biblioteca *Tkinter* de Python, con conexión a una base de datos MySQL para gestionar datos relacionados con hábitos de consumo y condiciones de salud de los usuarios.

La interfaz tiene una **estructura de formulario** donde se recopilan datos específicos mediante ‘*tk.Entry*’ y ‘*ttk.Combobox’*, así como un ‘*ttk.Treeview’* que permite visualizar en formato de tabla los datos almacenados en la base de datos. Cada sección de la interfaz está diseñada para facilitar la entrada de datos y la interacción con el usuario.

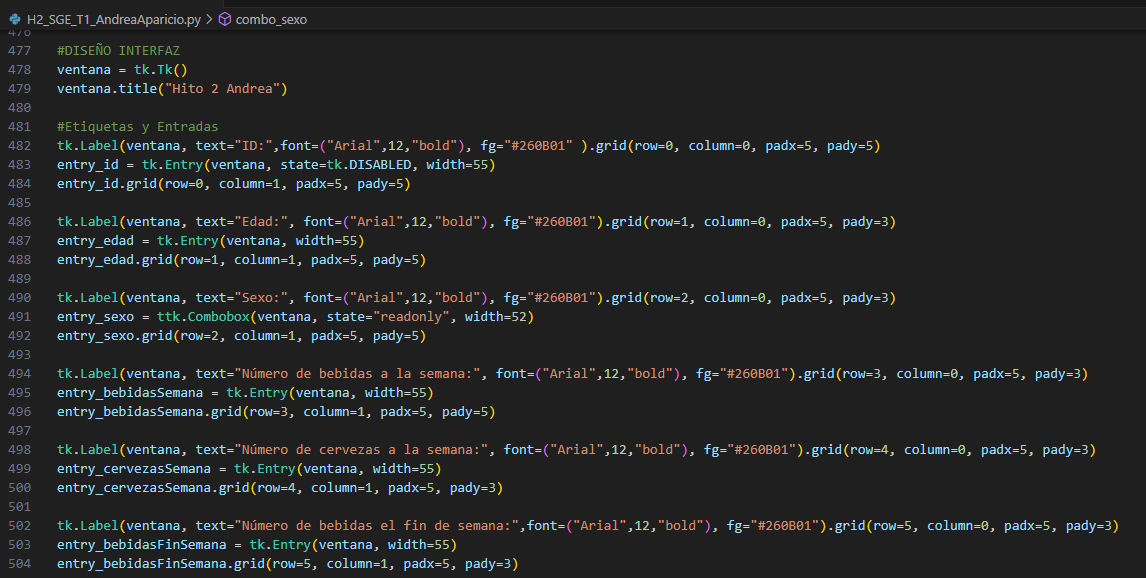
**Secciones de la Interfaz**

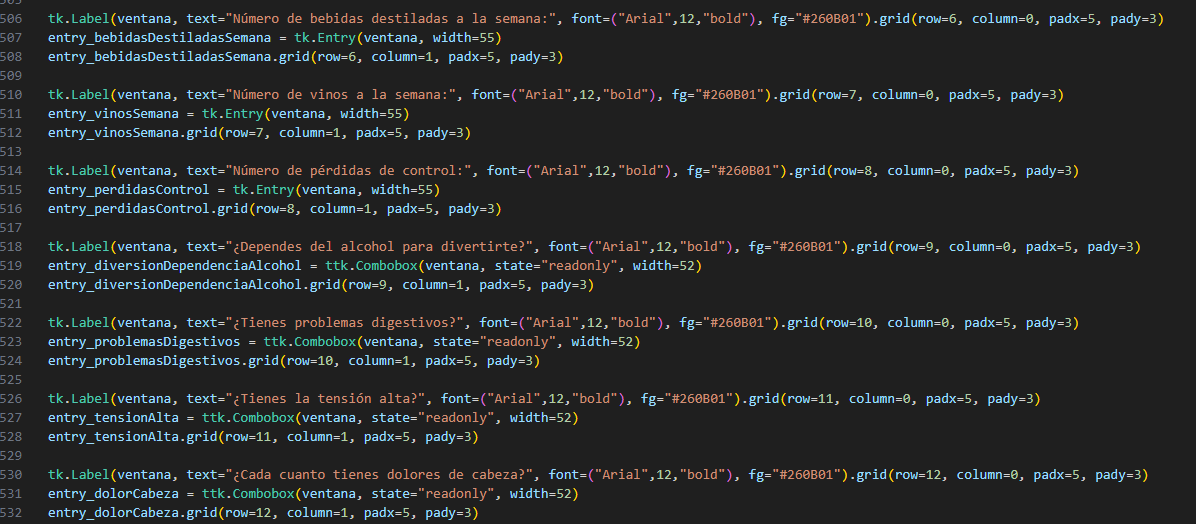
1. **Campos de Entrada**:
   * Se han definido varios campos para capturar diferentes atributos como Edad, Sexo, Número de bebidas a la semana, Número de cervezas a la semana, y otros.
   * La mayoría de los campos son de tipo ‘*tk.Entry’*, donde el usuario puede ingresar valores numéricos, y ‘*ttk.Combobox’* para opciones de selección predefinidas.
2. **Botones de Funcionalidad**:
   * **Ingresar**: Ejecuta la función *ingresar\_persona*, que añade nuevos registros a la base de datos.
   * **Actualizar tabla**: Llama a *actualizar\_persona*, actualizando los datos de la tabla en tiempo real.
   * **Eliminar**: Permite eliminar registros seleccionados en el Treeview.
   * **Limpiar**: Restablece los campos de entrada, eliminando cualquier dato que haya sido ingresado previamente.
   * **Filtrar**: Filtra los datos en el Treeview según ciertos criterios definidos.
   * **Exportar a Excel**: Guarda los datos actuales en un archivo Excel.
   * **Gráficos**: Los botones Gráfico circular y Gráfico barras generan gráficos estadísticos basados en los datos.
3. **Treeview**:
   * El ‘*ttk.Treeview’* está configurado para mostrar los datos de la base en formato de tabla, con columnas específicas para cada campo.
   * Nos muestra de manera estructurada todos los registros en la base de datos y permite seleccionar un registro para modificar o eliminar.
4. **Configuración y Distribución**:
   * Cada widget tiene configuraciones de fuente, color, y espaciado para mantener un diseño visual coherente.
   * La interfaz está adaptada para ser redimensionable, aprovechando el espacio en pantalla mediante *grid\_rowconfigure* y *grid\_columnconfigure*.

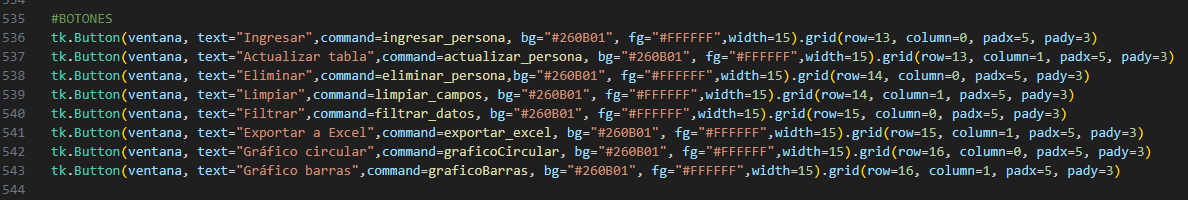
**2. Conexión a la Base de Datos MySQL**

La conexión a MySQL se realiza a través de la biblioteca *mysql.connector*, que permite ejecutar consultas SQL directamente desde el script de Python.

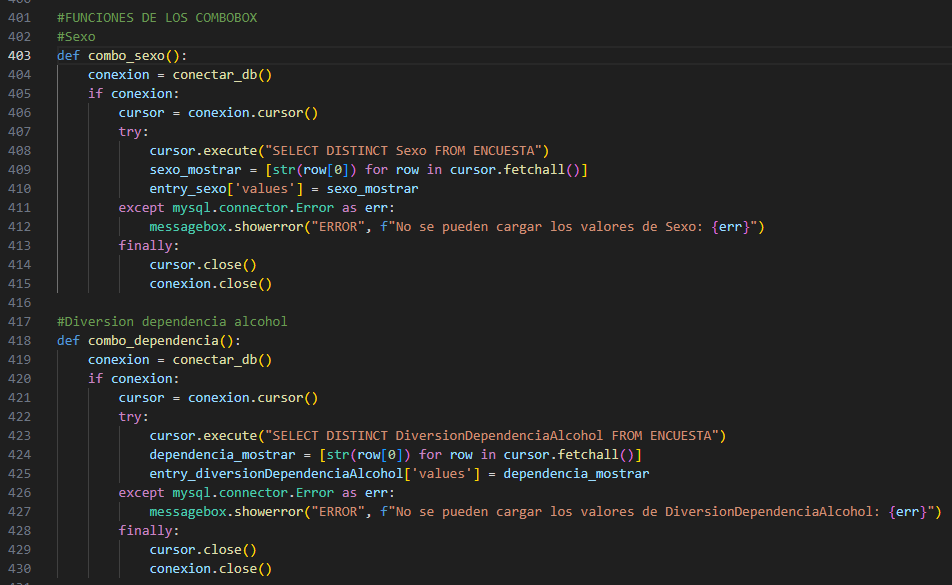
## Etiquetas y botones

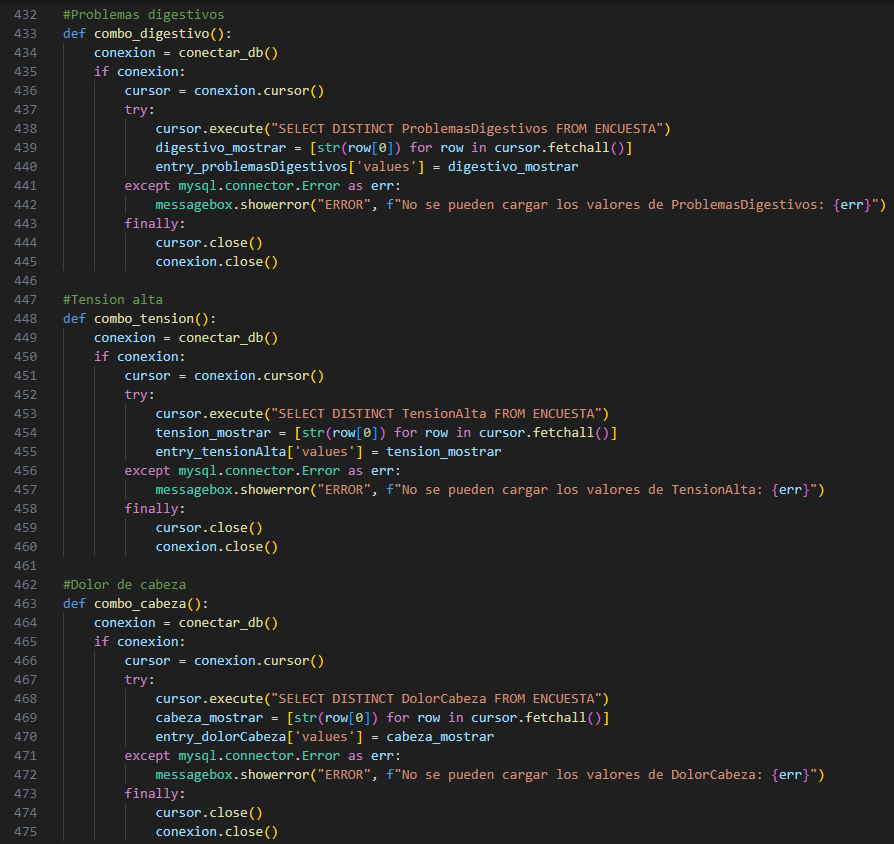






## Funciones ComboBox





Operaciones CRUD

Las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) implementadas en la interfaz gráfica de Tkinter nos permiten interactuar con una base de datos MySQL, gestionando la información de cada registro de manera eficiente.

## Ingresar

La función ‘*ingresar\_persona’* permite añadir nuevos registros a la tabla ‘ENCUESTA’. Esta operación se activa con el botón "**Ingresar**".

### Lógica

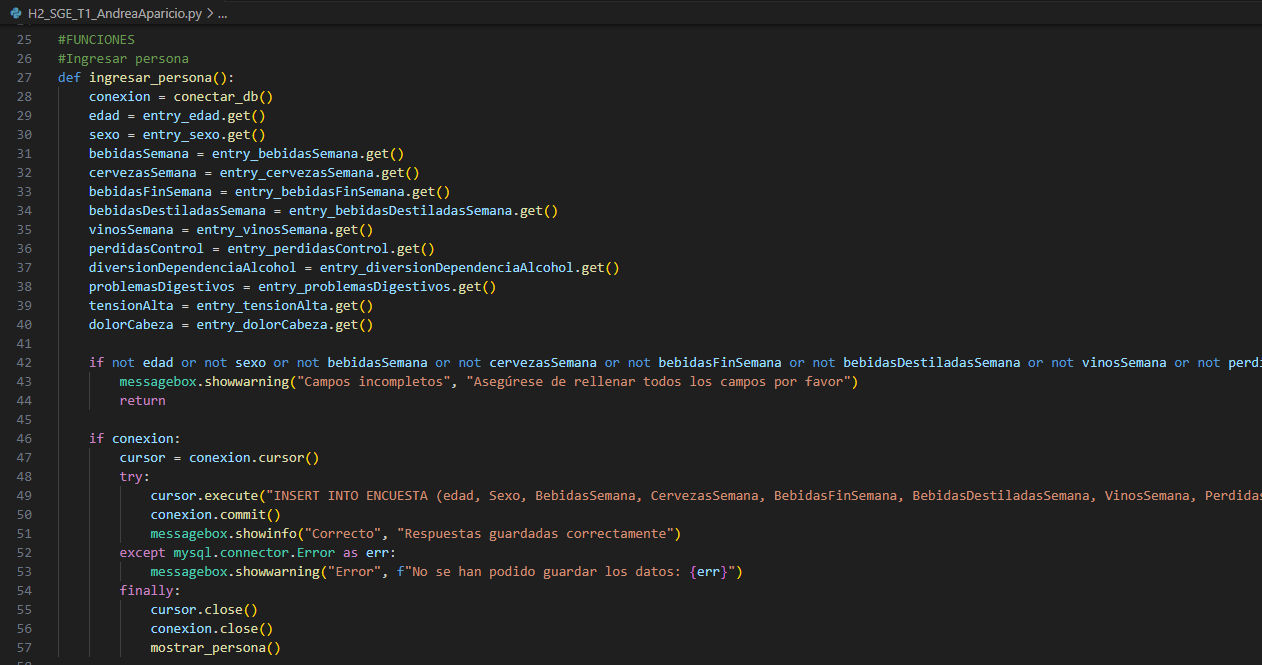
1. Se validan los campos de entrada para asegurarse de que ninguno esté vacío.

2. Se abre la conexión a la base de datos y se ejecuta la consulta SQL ‘INSERT INTO’ para añadir un nuevo registro.

3. Si la operación es exitosa, se muestra un mensaje de confirmación.

4. Al final, se cierra la conexión y se actualiza el Treeview para mostrar los cambios.

### Código



## Mostrar

La función ‘*mostrar\_persona’* permite visualizar todos los registros almacenados en la tabla ‘ENCUESTA’, mostrando los datos en el `Treeview` en la interfaz de usuario.

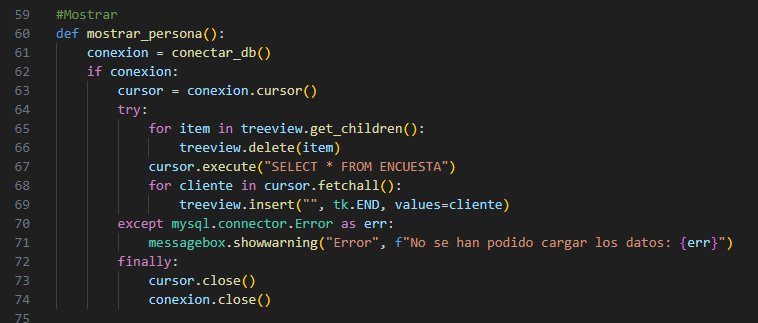
### Lógica

1. Se abre la conexión y se ejecuta la consulta ‘SELECT \* FROM ENCUESTA’.

2. Los resultados son insertados en el Treeview.

3. Se cierra la conexión al finalizar.

### Código



## Actualizar

La función ‘*actualizar\_persona’* permite modificar un registro existente, basándose en el **ID** seleccionado. Esta operación es activada por el botón "**Actualizar tabla**".

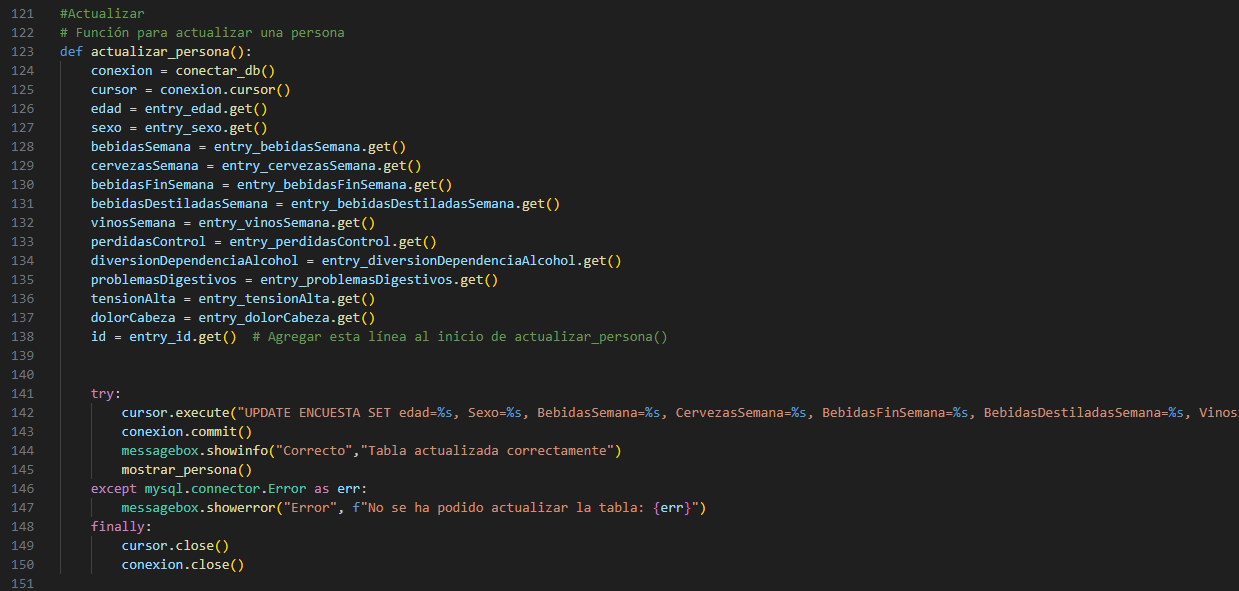
### Lógica

1. Se selecciona un registro en el Treeview y se cargan sus datos en los campos de entrada.

2. Al editar los valores y presionar "**Actualizar tabla**", la función realiza una consulta ‘UPDATE’ en la base de datos con los datos actualizados.

3. Se cierra la conexión y se muestra un mensaje de éxito si la operación fue correcta.

### Código



## Eliminar

La función ‘*eliminar\_persona’* permite borrar un registro seleccionado. Esta operación es activada por el botón "**Eliminar**".

### Lógica

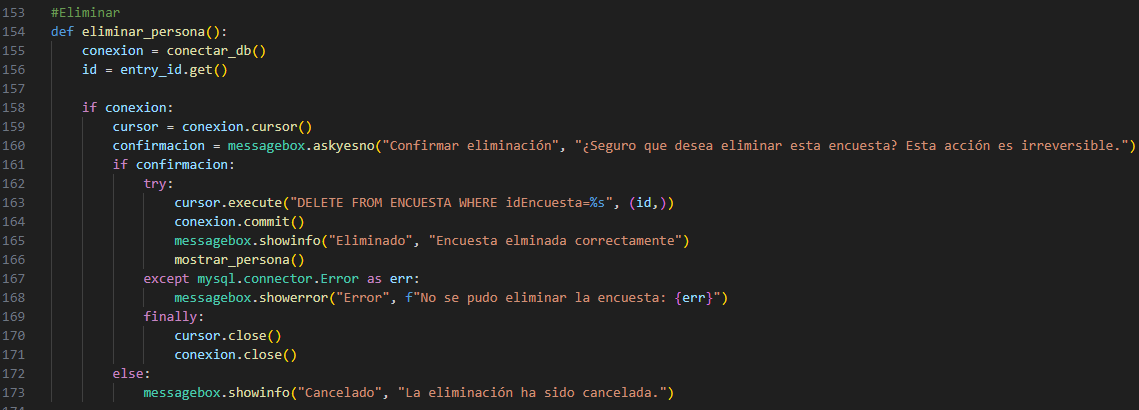
1. El usuario selecciona un registro en el Treeview.

2. Al presionar "**Eliminar**", aparece un cuadro de confirmación.

3. Si se confirma, se ejecuta una consulta ‘DELETE’ para borrar el registro en la base de datos.

4. Si la operación es exitosa, se actualiza el Treeview y se cierra la conexión.

Código



Consultas y Ordenación de Datos:

Para manejar la visualización y filtrado de registros en la aplicación de encuesta, se implementaron varias consultas y filtros en la interfaz. A continuación, se explican las funciones de consulta y filtrado, cómo se ordenan y muestran los resultados, y cómo se utilizan en la interfaz de usuario para seleccionar, limpiar, y filtrar registros en la base de datos MySQL.

## Seleccionar Persona

La función ‘*seleccionar\_persona’* permite al usuario seleccionar un registro en el Treeview para cargar sus datos en los campos de entrada, facilitando así su edición o eliminación.

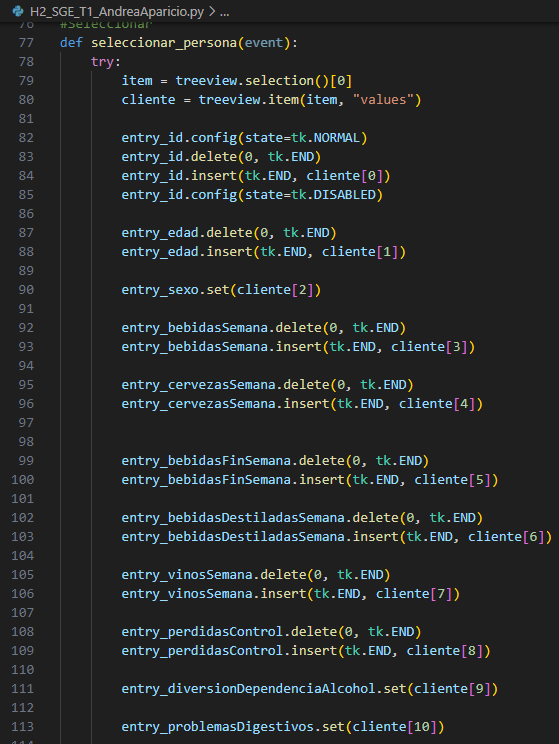
### Lógica

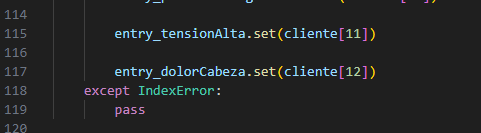
1. Esta función se activa mediante un evento de selección en el Treeview.

2. Obtiene el ID del registro seleccionado y, a partir de ello, carga todos los datos del registro en los campos de entrada para que puedan ser actualizados o visualizados.

3. El ID de la persona seleccionada se muestra en el campo correspondiente, pero se desactiva (‘DISABLED’) para evitar modificaciones accidentales del identificador único.

### Código





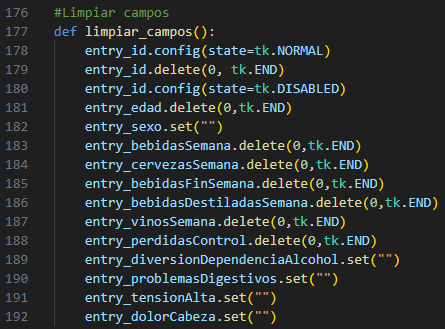
## Limpiar Campos

La función ‘*limpiar\_campos’* facilita el reinicio de los campos de entrada, eliminando cualquier valor visible. Esto es útil cuando el usuario desea realizar una nueva búsqueda o agregar un nuevo registro.

### Lógica

1. Al invocar esta función, todos los campos se limpian y el campo de ID se vuelve a desactivar (‘DISABLED’) para evitar entradas accidentales en ese campo.

### Código



## Filtrar Datos

La función ‘*filtrar\_dato*s’ permite al usuario aplicar filtros específicos para buscar registros en la base de datos, utilizando diversos criterios como edad, sexo, frecuencia de bebidas, entre otros.

### Lógica

1. Se construye una consulta SQL dinámica. La base de la consulta es ‘SELECT ... FROM ENCUESTA WHERE 1=1’.

2. Para cada campo de entrada con un valor ingresado, se agrega una condición ‘AND’ correspondiente, ajustando el filtro de búsqueda.

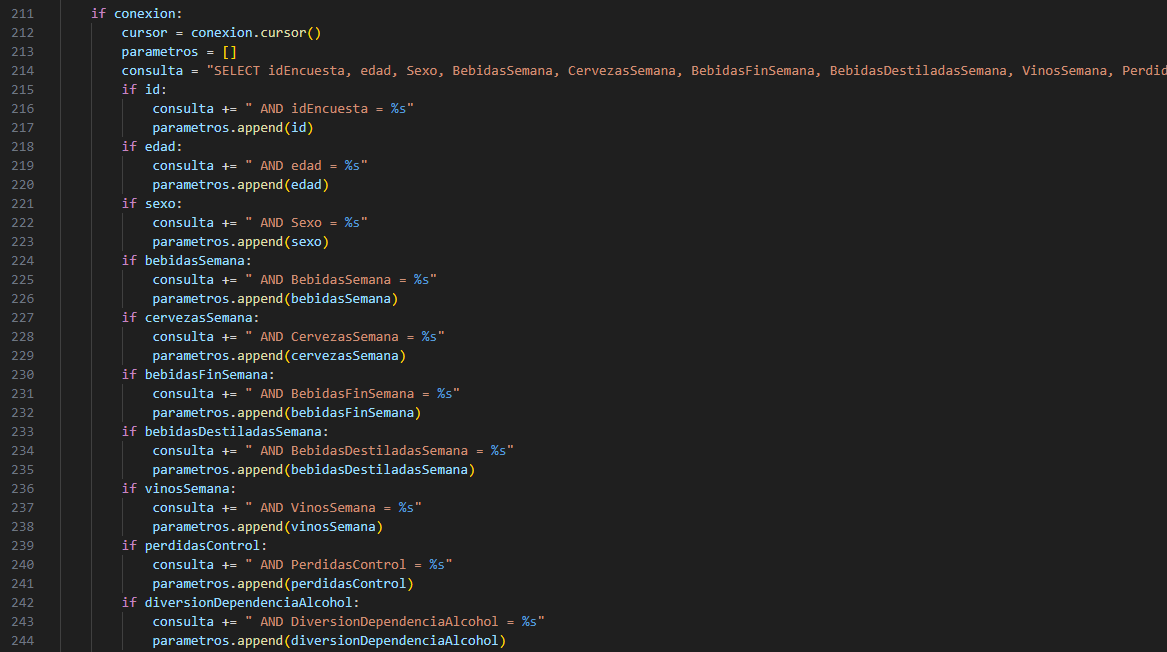
3. Se ejecuta la consulta con los parámetros correspondientes y se muestran los resultados en el Treeview.

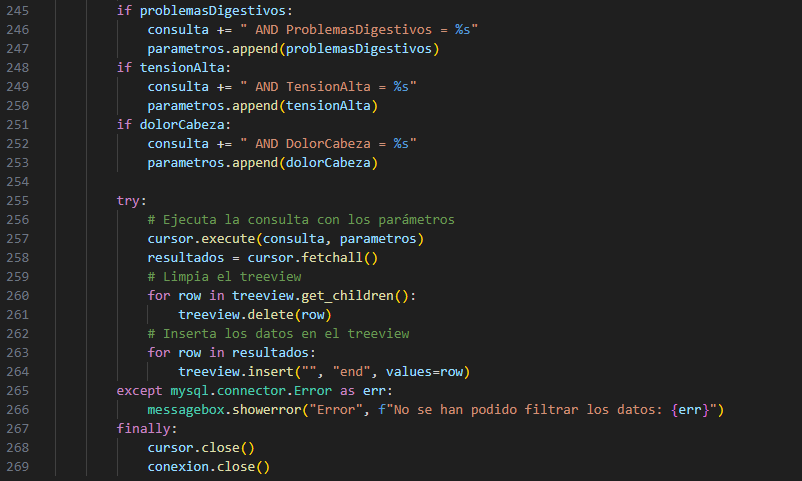
**Ejemplo de Uso**:

Si el usuario desea ver los registros donde la ‘edad’ sea “25” y el ‘sexo’ sea "Femenino". Solo necesita ingresar esos datos y hacer clic en "**Filtrar**", lo que activará esta función para aplicar los filtros específicos.

### Código







- La consulta ‘SELECT ... FROM ENCUESTA WHERE 1=1’ se utiliza como base para **agregar condiciones ‘AND’ en función de los campos rellenados por el usuario**. Esto permite realizar consultas flexibles sin modificar la estructura de la función cada vez.

- **Parámetros de consulta**: Al utilizar una lista ‘parametros’, los valores ingresados se pasan de manera segura mediante el uso de ‘execute(consulta, parametros)’, evitando problemas de inyección SQL.

Después de aplicar los filtros, los datos se muestran en el Treeview, donde cada columna representa un campo del registro.

Exportación de Resultados a Excel

Este código permite exportar los datos visualizados en un componente Treeview de una interfaz gráfica a un archivo Excel. Los datos se almacenan en un ‘*DataFrame’* de pandas y se guardan en formato ‘.xlsx’ usando la biblioteca *openpyxl*.

### Lógica

**1.** **Obtención de Datos del Treeview**:

* Se crea una lista vacía llamada “rows”.
* Mediante un ciclo **for**, se itera sobre cada elemento en el Treeview, extrayendo sus valores y agregándolos a “rows”.
* Si no hay datos en Treeview, se muestra un mensaje de advertencia indicando que no hay datos para exportar.

**2. Creación del ‘*DataFrame’*:**

* Se define una lista “columnas” que contiene los nombres de cada columna que aparecerán en el archivo Excel.
* Se crea un ‘*DataFrame’* de pandas a partir de la lista “rows”, utilizando “columnas” como los encabezados.

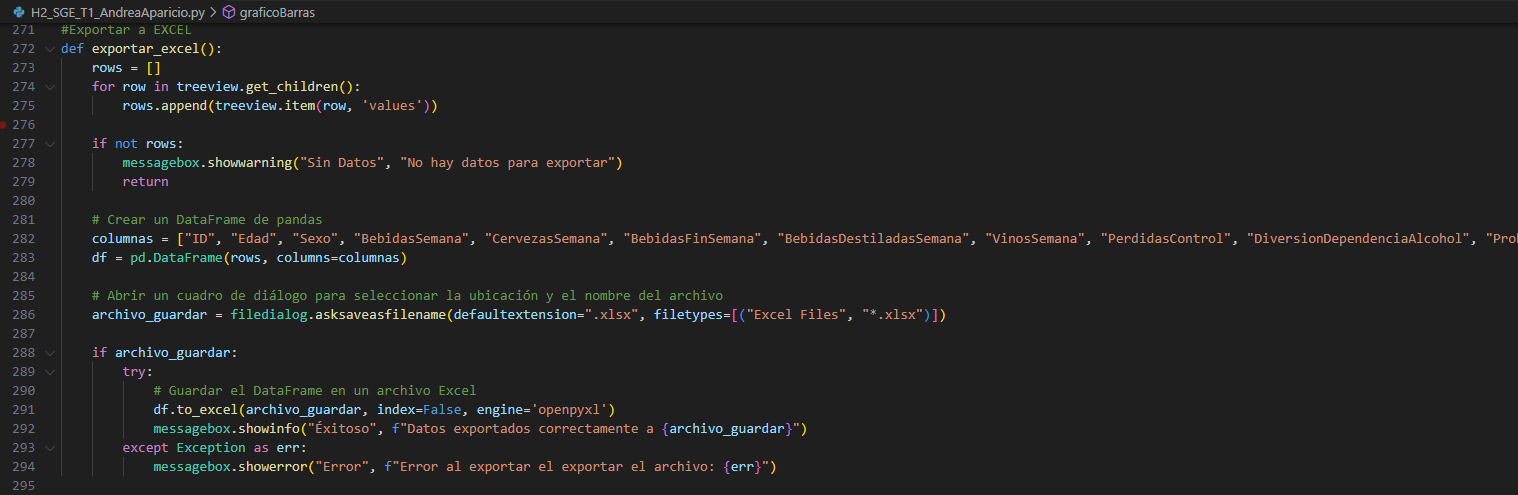
**3. Selección del Archivo y Ubicación de Guardado:**

* A través del diálogo *asksaveasfilename* de ‘filedialog’, se permite al usuario **elegir la ubicación y el nombre del archivo** a exportar. Este diálogo asegura que el archivo se guarde con la extensión ‘.xlsx’.

**4. Exportación a Excel:**

* Si el usuario selecciona un nombre y ubicación, se intenta guardar el ‘*DataFrame’* en un archivo ‘.xlsx’ usando ‘to\_excel’, especificando el motor ‘openpyxl’.
* En caso de éxito, se muestra un **mensaje informativo** de que la exportación fue exitosa.
* Si ocurre algún error durante la exportación, se muestra un **mensaje de error** detallando la causa.

### Código



Visualización de Datos en Gráficos

Los gráficos permiten visualizar de manera clara y comprensible el consumo de bebidas según los datos almacenados en la base de datos.

## Gráfico de Barras

Este gráfico muestra la **distribución del consumo total de bebidas** por **grupo de edad**. Representa de manera comparativa el consumo acumulado de diferentes tipos de bebidas (cervezas, vinos, bebidas destiladas, etc.) en rangos de edades específicos.

### Personalización

**1.** **Estilo y Color**:

* Uso de la paleta de colores ‘Set2’ de *seaborn*.
* Configuración de un tamaño de figura adecuado (‘figsize = (10, 6)’) para buena visibilidad.

**2. Etiquetas y Títulos:**

* **Título descriptivo**: "Distribución de Consumo de Bebidas por Grupo de Edad".
* Etiquetas en los ejes para que no haya confusiones.

**3. Datos Filtrados:**

* **Se eliminan filas con valores nulos o cero** en las columnas de consumo para evitar distorsiones en la gráfica.

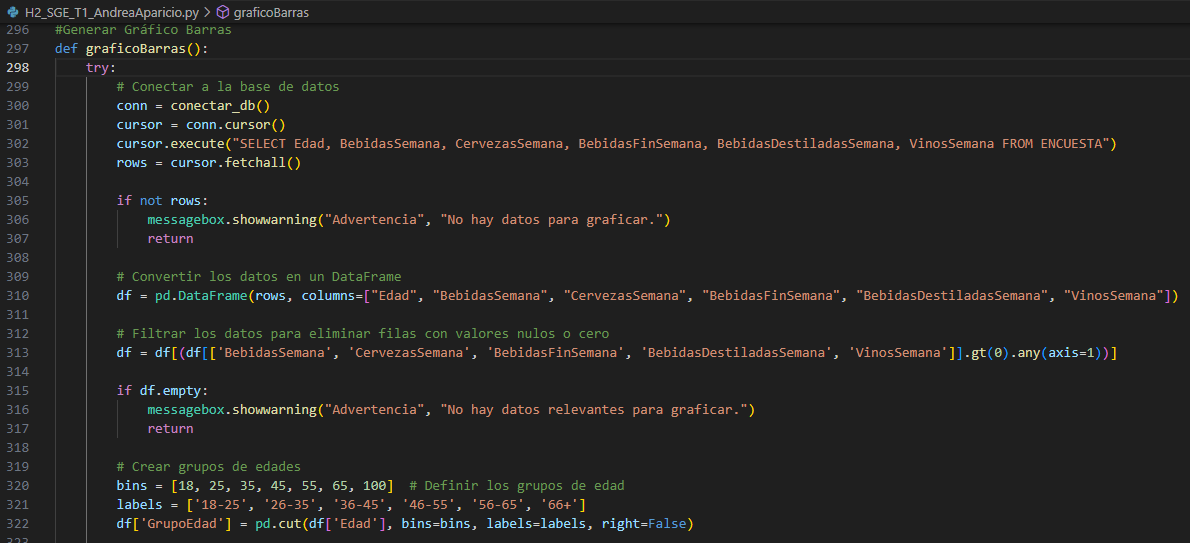
**4. Rotación de Etiquetas:**

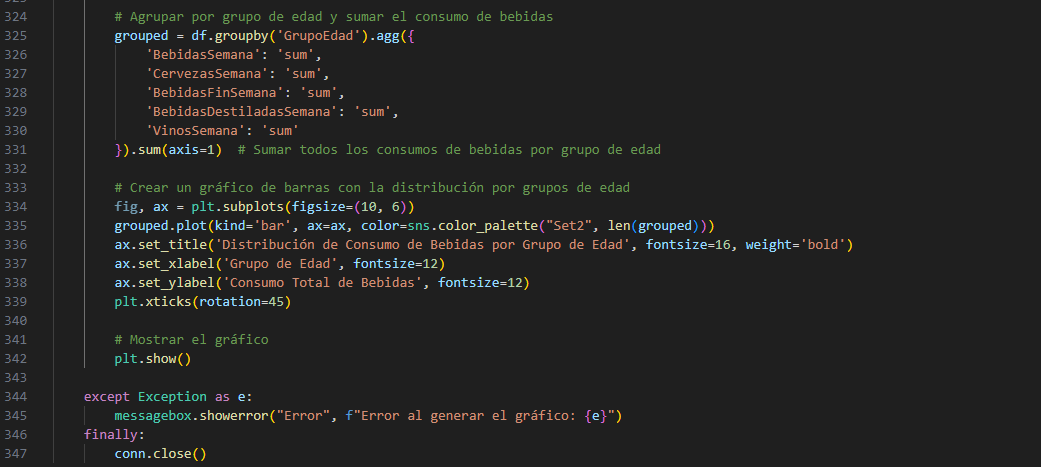
* Rotación de 45° en las etiquetas del eje X para facilitar la lectura.

### Explicación del Gráfico

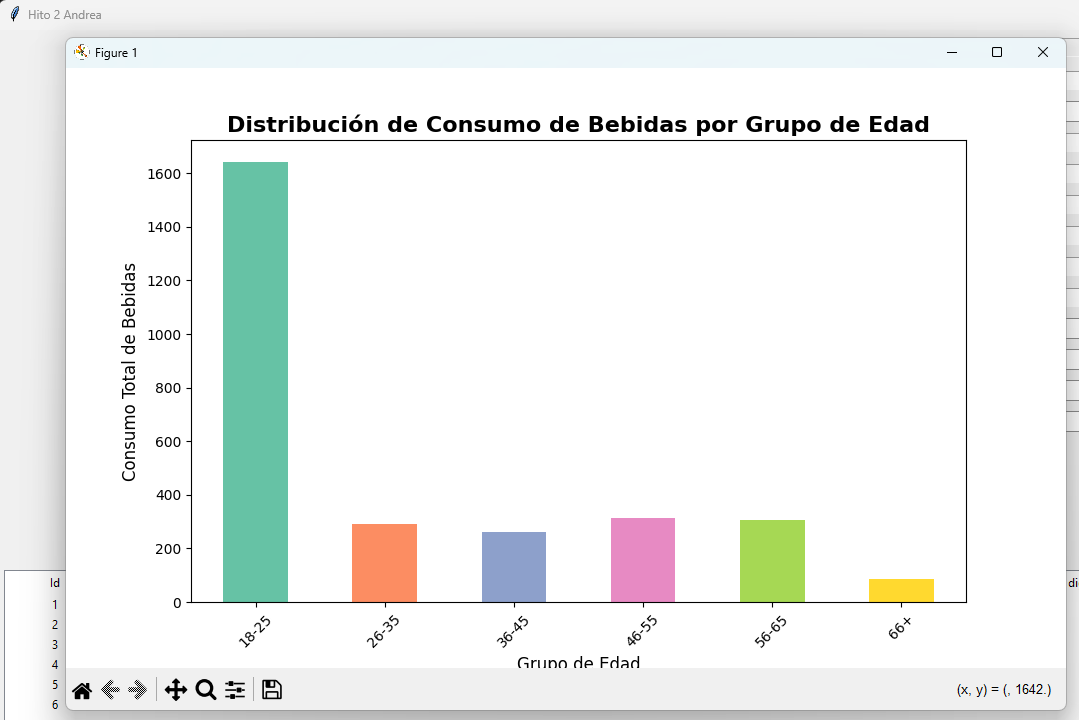
* El **eje X** muestra los grupos de edad (por ejemplo, 18-25, 26-35, etc.).
* El **eje Y** representa el consumo total de bebidas.
* La **altura de cada** **barra** indica el nivel de consumo relativo en cada grupo de edad.

### Código





### Resultado



## Gráfico Circular

Este gráfico representa la **proporción del consumo de bebidas** en cada grupo de edad. Es ideal para identificar visualmente qué grupo contribuye más o menos al consumo total.

### Personalización

**1. Estilo y Color:**

* Uso de la paleta de colores ‘Set2’ de *seaborn* para colores distinguibles.
* Ajuste del gráfico para que sea un círculo perfecto.

**2. Etiquetas y Títulos:**

* **Título descriptivo**: "Distribución de Consumo de Bebidas por Grupo de Edad".
* Etiquetas con el porcentaje de consumo total para cada segmento.

**3. Datos Filtrados**:

* Al igual que en el gráfico de barras, **se eliminan filas con valores nulos o cero**.

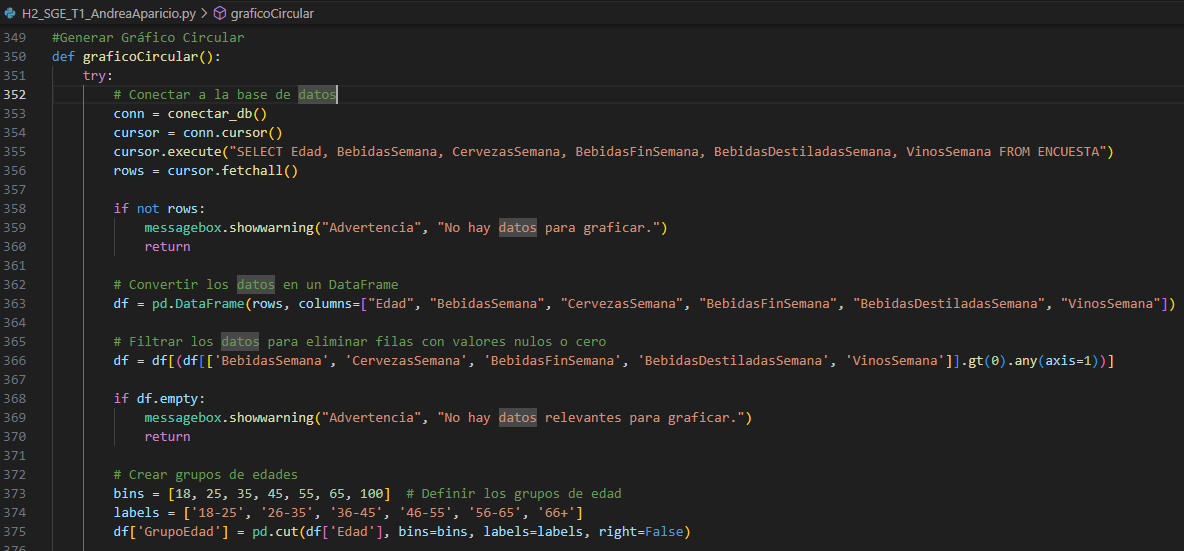
**4. Autopct:**

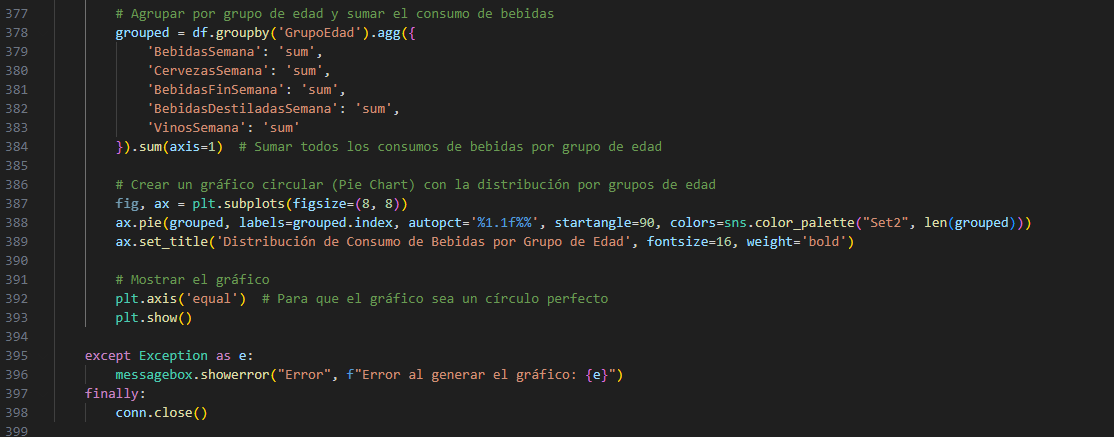
* El parámetro *autopct='%1.1f%%'* agrega el porcentaje con un decimal en cada segmento.

### Explicación del Gráfico

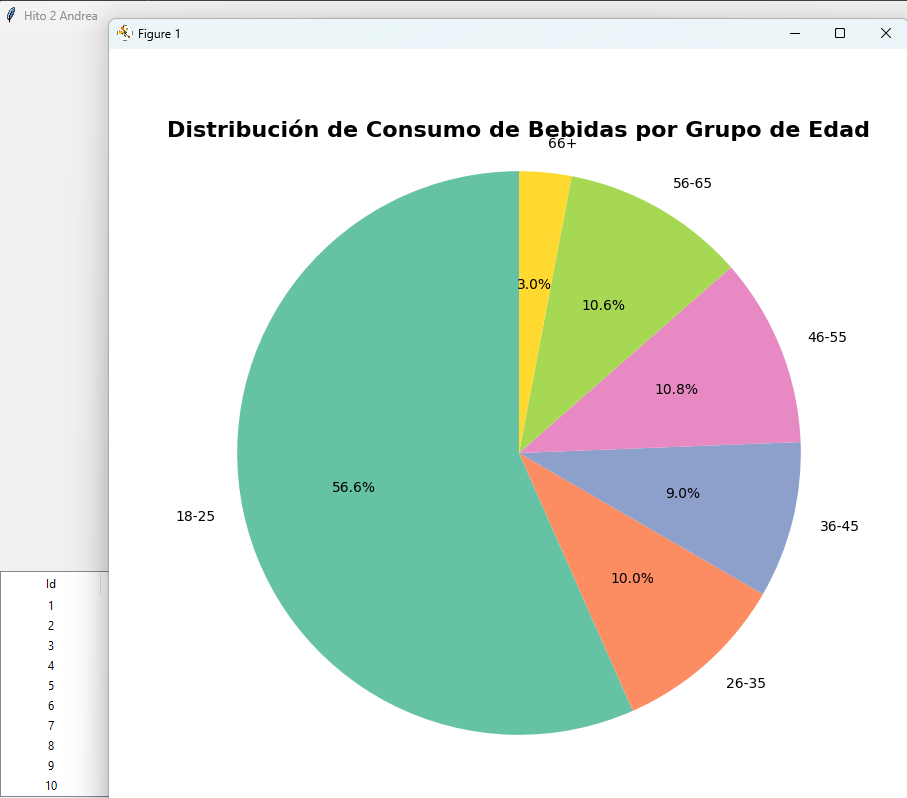
* Cada segmento del círculo representa un grupo de edad.
* El tamaño relativo del segmento refleja la proporción del consumo total de bebidas.
* Es útil para destacar visualmente qué grupo consume más bebidas en comparación con los demás.

### Código





### Resultado



Código comentado

Se espera que el código del proyecto esté comentado, explicando las partes clave en el repositorio de GitHub.