```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tkinter import (
    Tk, filedialog, Toplevel, Text, Scrollbar, Frame, Button,
Checkbutton,
    BooleanVar, Label, Listbox, MULTIPLE, END, BOTH, Canvas, LEFT, RIGHT,
Y, VERTICAL,
    LabelFrame  # Added LabelFrame to imports
)

# Global variable to store analysis results

# Global variable to store analysis results
analysis_results = {}

# Load and clean data dynamically
def adat_beolvasas():
    """
    Load data from a selected file (CSV, Excel, or TXT).
    """
    fajl_nev = filedialog.askopenfilename(
        title="Válassz egy fájlt",
        filetypes=[
            ("Minden adatfájl", "*.csv;*.xlsx;*.xls;*.txt"),
            ("CSV fájlok", "*.csv"),
            ("Excel fájlok", "*.xlsx;*.xls"),
            ("TXT fájlok", "*.txt"),
        ],
    )
    if not fajl_nev:
        return None

    try:
        if fajl_nev.endswith(".csv"):
            adat = pd.read_csv(fajl_nev)
        elif fajl_nev.endswith((".xls", ".xlsx")):
            adat = pd.read_excel(fajl_nev, engine="openpyxl")
        elif fajl_nev.endswith(".txt"):
            with open(fajl_nev, 'r') as file:
                first_line = file.readline()
                delimiter = detect_delimiter(first_line)
                adat = pd.read_csv(fajl_nev, delimiter=delimiter)
        else:
            raise ValueError("Nem támogatott fájlformátum!")
        return adat
    except Exception as e:
        megjelenit_ablak("Hiba", f"Hiba történt az állomány beolvasása
során: {e}")
        return None

# Detect delimiter for TXT files
def detect_delimiter(first_line):
    delimiters = [',', '\t', ';', '|']
    delimiter_count = {delimiter: first_line.count(delimiter) for
delimiter in delimiters}
    return max(delimiter_count, key=delimiter_count.get)
```

```
# Statistical analysis selection window
def select_analysis_options(adatok):
    """
    Creates a GUI for the user to choose statistical tools and columns
for analysis.
    """
    analysis_window = Toplevel()
    analysis_window.title("Analysis Options")
    analysis_window.geometry("600x600")  # Increased height to
accommodate more options

    # Column selection
    Label(analysis_window, text="Válassza ki az elemzendő
oszlopokat:").pack()
    column_frame = Frame(analysis_window)
    column_frame.pack()

    column_vars = {}
    for col in adatok.columns:
        var = BooleanVar(value=True)
        column_vars[col] = var
        Checkbutton(column_frame, text=col,
variable=var).pack(anchor='w')

    # Add Select All and Select None buttons for columns
    def select_all_columns():
        for var in column_vars.values():
            var.set(True)

    def select_none_columns():
        for var in column_vars.values():
            var.set(False)

    Button(analysis_window, text="Mindet kijelöli",
command=select_all_columns).pack(pady=5)
    Button(analysis_window, text="Egyiket sem jelöli ki",
command=select_none_columns).pack(pady=5)

    # Statistical tools selection
    Label(analysis_window, text="Válassza ki az alkalmazandó statisztikai
eszközöket:").pack()
    stats_frame = Frame(analysis_window)
    stats_frame.pack()

    stats_tools = {
        "Átlag (Mean)": BooleanVar(value=True),
        "Medián (Median)": BooleanVar(value=True),
        "Szórás (Standard Deviation)": BooleanVar(value=True),
        "Minimum": BooleanVar(value=True),
        "Maximum": BooleanVar(value=True),
        "Módusz (Mode)": BooleanVar(value=True),
        "Gyakoriság (Value Counts)": BooleanVar(value=True),
    }
    for tool, var in stats_tools.items():
        Checkbutton(stats_frame, text=tool,
variable=var).pack(anchor='w')

    # Confirm button
    Button(
        analysis_window,
        text="Elemzés indítása",
        command=lambda: perform_analysis(adatok, column_vars,
stats_tools),
    ).pack(pady=10)

    # Close button with graphical window callback
    Button(
        analysis_window,
        text="Tovább a vizualizációhoz",
        command=lambda: [select_visualization_options(adatok),
analysis_window.destroy()],
    ).pack()

    analysis_window = Toplevel()
    analysis_window.title("Elemzési Opcók")
    Button(analysis_window, text="Halmozott Sávdiagram", command=lambda:
perform_visualization(adatok, column_vars, {}, {}, {})).pack(pady=5)
    Button(analysis_window, text="Optimalizált Hőtérkép", command=lambda:
perform_visualization(adatok, column_vars, {}, {}, {})).pack(pady=5)

    print("Analysis Options Window Created.")

# Perform the selected analysis
def perform_analysis(adatok, column_vars, stats_tools):
    """
    Perform the analysis based on selected options.
    """
    global analysis_results
    analysis_results = {}
    # Filter columns
    selected_columns = [col for col, var in column_vars.items() if
var.get()]
    data_to_analyze = adatok[selected_columns]

    results = ""

    # Numerical analysis
    numeric_data = data_to_analyze.select_dtypes(include=["number"])
    if not numeric_data.empty:
        results += "Numerikus adatok statisztikai eredményei:\n\n"
        analysis_results['numeric'] = {}
        if stats_tools["Átlag (Mean)"].get():
            mean_result = numeric_data.mean()
            results += f"Átlag:\n{mean_result}\n\n"
            analysis_results['numeric']['mean'] = mean_result
        if stats_tools["Medián (Median)"].get():
            median_result = numeric_data.median()
            results += f"Medián:\n{median_result}\n\n"
            analysis_results['numeric']['median'] = median_result
        if stats_tools["Szórás (Standard Deviation)"].get():
            std_result = numeric_data.std()
            results += f"Szórás:\n{std_result}\n\n"
            analysis_results['numeric']['std'] = std_result
        if stats_tools["Minimum"].get():
            min_result = numeric_data.min()
            results += f"Minimum:\n{min_result}\n\n"
            analysis_results['numeric']['min'] = min_result
        if stats_tools["Maximum"].get():
            max_result = numeric_data.max()
            results += f"Maximum:\n{max_result}\n\n"
            analysis_results['numeric']['max'] = max_result

    # Categorical analysis
    categorical_data = data_to_analyze.select_dtypes(include=["object",
"category"])
    if not categorical_data.empty:
        results += "Kategorikus adatok statisztikai eredményei:\n\n"
        analysis_results['categorical'] = {}
        if stats_tools["Módusz (Mode)"].get():
            mode_result = categorical_data.mode().iloc[0]
            results += f"Módusz:\n{mode_result}\n\n"
            analysis_results['categorical']['mode'] = mode_result
        if stats_tools["Gyakoriság (Value Counts)"].get():
            analysis_results['categorical']['value_counts'] = {}
            for col in categorical_data.columns:
                value_counts = categorical_data[col].value_counts()
                results += f"Gyakoriság - {col}:\n{value_counts}\n\n"
                analysis_results['categorical']['value_counts'][col] =
value_counts

    if results:
        megjelenit_ablak("Elemzés Eredményei", results)
    else:
        megjelenit_ablak("Elemzés Eredményei", "Nincs megjeleníthető
eredmény a kiválasztott opciókhoz.")

# Display results in a tkinter window
def megjelenit_ablak(cim, szoveg):
    ablak = Toplevel()
    ablak.title(cim)

    frame = Frame(ablak)
    frame.pack(fill="both", expand=True)

    szovegdoboz = Text(frame, wrap="word", bg="white", fg="black")
    szovegdoboz.insert("1.0", szoveg)
    szovegdoboz.config(state="disabled")
    szovegdoboz.pack(side="left", fill="both", expand=True)

    scrollbar = Scrollbar(frame, command=szovegdoboz.yview)
    scrollbar.pack(side="right", fill="y")
    szovegdoboz.config(yscrollcommand=scrollbar.set)

    Button(ablak, text="Bezárás", command=ablak.destroy).pack()


def select_visualization_options(adatok):
    """
    Creates a GUI for the user to choose data visualization options.
    """
    print("Opening Visualization Options Window...")  # Debugging log

      # Create the Toplevel window
    visualization_window = Toplevel()
    visualization_window.title("Visualization Options")
    visualization_window.geometry("800x600")  # Adjusted size
    visualization_window.resizable(True, True)  # Allow resizing

    # Create a main frame to hold all content
    main_frame = Frame(visualization_window)
    main_frame.pack(fill=BOTH, expand=1)

    # Create a canvas to allow scrolling
    canvas = Canvas(main_frame)
    canvas.pack(side=LEFT, fill=BOTH, expand=1)

    # Add a scrollbar to the canvas
    scrollbar = Scrollbar(main_frame, orient=VERTICAL,
command=canvas.yview)
    scrollbar.pack(side=RIGHT, fill=Y)

    # Configure the canvas
    canvas.configure(yscrollcommand=scrollbar.set)
    canvas.bind('<Configure>', lambda e:
canvas.configure(scrollregion=canvas.bbox("all")))

    # Create a frame inside the canvas
    content_frame = Frame(canvas)
    canvas.create_window((0, 0), window=content_frame, anchor="nw")

    # Organize the content_frame using grid layout
    # Create frames for individual and combined plots
    individual_frame = LabelFrame(content_frame, text="Egyéni Diagramok")
    combined_frame = LabelFrame(content_frame, text="Kombinált
Diagramok")

    # Place the frames side by side
    individual_frame.grid(row=0, column=0, padx=10, pady=10,
sticky="nsew")
    combined_frame.grid(row=0, column=1, padx=10, pady=10, sticky="nsew")

    # Configure grid weights to allow frames to expand
    content_frame.grid_columnconfigure(0, weight=1)
    content_frame.grid_columnconfigure(1, weight=1)

    # Individual Plots Section
    # Column selection for individual plots
    Label(individual_frame, text="Válassza ki a megjelenítendő
oszlopokat:").pack()
    columns_frame = Frame(individual_frame)
    columns_frame.pack()

    column_vars = {}
    for col in adatok.columns:
        var = BooleanVar(value=True)
        column_vars[col] = var
        Checkbutton(columns_frame, text=col,
variable=var).pack(anchor='w')

    # Add Select All and Select None buttons for individual columns
    def select_all_individual_columns():
        for var in column_vars.values():
            var.set(True)

    def select_none_individual_columns():
        for var in column_vars.values():
            var.set(False)

    buttons_frame_individual = Frame(individual_frame)
    buttons_frame_individual.pack(pady=5)
    Button(buttons_frame_individual, text="Mindet kijelöli",
command=select_all_individual_columns).pack(side='left', padx=5)
    Button(buttons_frame_individual, text="Egyiket sem jelöli ki",
command=select_none_individual_columns).pack(side='left', padx=5)

    # Visualization type selection for individual plots
    Label(individual_frame, text="Válassza ki a diagram
típusát:").pack(pady=10)
    vis_types_frame = Frame(individual_frame)
    vis_types_frame.pack()

    visualization_types = {
        "Hisztogram (Numeric)": BooleanVar(value=True),
        "Boxplot (Numeric)": BooleanVar(value=True),
        "Oszlopdiagram (Categorical)": BooleanVar(value=True),
        "Kördiagram (Pie Chart - Categorical)": BooleanVar(value=True),
        "Statisztikák Diagramja (Statistics Charts)":
BooleanVar(value=False),
    }
    for vis_type, var in visualization_types.items():
        Checkbutton(vis_types_frame, text=vis_type,
variable=var).pack(anchor='w')

    # Combined Plots Section
    # Column selection for combined plots
    Label(combined_frame, text="Válassza ki a kombinált
oszlopokat:").pack()
    combined_list_frame = Frame(combined_frame)
    combined_list_frame.pack()

    combined_columns_listbox = Listbox(combined_list_frame,
selectmode=MULTIPLE, exportselection=0, height=10)
    combined_columns_listbox.pack(side=LEFT, fill=BOTH, expand=True)
    combined_scrollbar = Scrollbar(combined_list_frame, orient=VERTICAL)
    combined_scrollbar.config(command=combined_columns_listbox.yview)
    combined_scrollbar.pack(side=RIGHT, fill=Y)

combined_columns_listbox.config(yscrollcommand=combined_scrollbar.set)

    for idx, col in enumerate(adatok.columns):
        combined_columns_listbox.insert(END, col)

    # Add Select All and Select None buttons for combined columns
    def select_all_combined_columns():
        combined_columns_listbox.select_set(0, END)

    def select_none_combined_columns():
        combined_columns_listbox.select_clear(0, END)

    buttons_frame_combined = Frame(combined_frame)
    buttons_frame_combined.pack(pady=5)
    Button(buttons_frame_combined, text="Mindet kijelöli",
command=select_all_combined_columns).pack(side='left', padx=5)
    Button(buttons_frame_combined, text="Egyiket sem jelöli ki",
command=select_none_combined_columns).pack(side='left', padx=5)

    # Visualization types for combined plots
    Label(combined_frame, text="Válassza ki a kombinált diagram
típusát:").pack(pady=10)
    combined_vis_types_frame = Frame(combined_frame)
    combined_vis_types_frame.pack()

    combined_visualization_types = {
        "Szórásdiagram (Scatter Plot)": BooleanVar(value=False),
        "Páros Diagram (Pair Plot)": BooleanVar(value=False),
        "Hőtérkép (Heatmap)": BooleanVar(value=False),
        "Boxplot Kategória szerint": BooleanVar(value=False),
        "Csoportosított Oszlopdiagram": BooleanVar(value=False),
        "Két Kategorikus Változó Diagramja": BooleanVar(value=False),  #
New option added
    }
    for vis_type, var in combined_visualization_types.items():
        Checkbutton(combined_vis_types_frame, text=vis_type,
variable=var).pack(anchor='w')

    # Confirm and Close buttons
    buttons_frame = Frame(content_frame)
    buttons_frame.grid(row=1, column=0, columnspan=2, pady=10)
    Button(
        buttons_frame,
        text="Megjelenítés indítása",
        command=lambda: perform_visualization(
            adatok, column_vars, visualization_types,
            combined_columns_listbox, combined_visualization_types
        ),
    ).pack(side='left', padx=5)
    Button(buttons_frame, text="Bezárás",
command=visualization_window.destroy).pack(side='left', padx=5)

    # Ensure the window is displayed and responsive
    print("Visualization Options Window Ready.")  # Debugging log


def perform_visualization(adatok, column_vars, visualization_types,
combined_columns_listbox, combined_visualization_types):
    """
    Perform visualization based on selected options.
    """
    global analysis_results
    # Filter selected columns
    selected_columns = [col for col, var in column_vars.items() if
var.get()]
    data_to_visualize = adatok[selected_columns]

    # Generate individual visualizations
    if visualization_types["Hisztogram (Numeric)"].get():
        numeric_data =
data_to_visualize.select_dtypes(include=["number"])
```

```
        if not numeric_data.empty:
            for col in numeric_data.columns:
                plt.figure(figsize=(10,6))
                sns.histplot(numeric_data[col], kde=True)
                plt.title(f"Hisztogram: {col}")
                if 'mean' in analysis_results.get('numeric', {}) and col
in analysis_results['numeric']['mean']:
                    plt.axvline(analysis_results['numeric']['mean'][col],
color='r', linestyle='dashed', linewidth=2, label='Átlag')
                if 'median' in analysis_results.get('numeric', {}) and
col in analysis_results['numeric']['median']:

plt.axvline(analysis_results['numeric']['median'][col], color='g',
linestyle='dotted', linewidth=2, label='Medián')
                plt.legend()
                plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "Nincs numerikus adat a
hisztogramokhoz.")

    if visualization_types["Boxplot (Numeric)"].get():
        numeric_data =
data_to_visualize.select_dtypes(include=["number"])
        if not numeric_data.empty:
            plt.figure(figsize=(10,6))
            sns.boxplot(data=numeric_data)
            plt.title("Boxplot a numerikus oszlopokhoz")
            plt.xticks(rotation=45)
            plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "Nincs numerikus adat a
boxplotokhoz.")

    if visualization_types["Oszlopdiagram (Categorical)"].get():
        categorical_data =
data_to_visualize.select_dtypes(include=["object", "category"])
        if not categorical_data.empty:
            for col in categorical_data.columns:
                if categorical_data[col].nunique() < 20:  # Limit to
manageable categories
                    plt.figure(figsize=(10,6))
                    sns.countplot(y=col, data=categorical_data,
order=categorical_data[col].value_counts().index)
                    plt.title(f"Oszlopdiagram: {col}")
                    plt.xlabel("Előfordulások száma")
                    plt.ylabel(col)
                    plt.show()
                else:
                    megjelenit_ablak("Figyelmeztetés", f"Túl sok
kategória az oszlopdiagramhoz: {col}")
        else:
            megjelenit_ablak("Figyelmeztetés", "Nincs kategorikus adat az
oszlopdiagramokhoz.")

    if visualization_types["Kördiagram (Pie Chart - Categorical)"].get():
        categorical_data =
data_to_visualize.select_dtypes(include=["object", "category"])
        if not categorical_data.empty:
            for col in categorical_data.columns:
                if categorical_data[col].nunique() < 10:  # Limit to
manageable categories
                    plt.figure(figsize=(8,8))
                    categorical_data[col].value_counts().plot(kind="pie",
autopct='%1.1f%%')
                    plt.title(f"Kördiagram: {col}")
                    plt.ylabel('')  # Hide y-label for pie chart
                    plt.show()
                else:
                    megjelenit_ablak("Figyelmeztetés", f"Túl sok
kategória a kördiagramhoz: {col}")
        else:
            megjelenit_ablak("Figyelmeztetés", "Nincs kategorikus adat a
kördiagramokhoz.")

    if visualization_types["Statisztikák Diagramja (Statistics
Charts)"].get():
        # Visualize numerical statistics
        if 'numeric' in analysis_results:
            numeric_stats = analysis_results['numeric']
            for stat_name, stat_values in numeric_stats.items():
                plt.figure(figsize=(10,6))
                stat_values.plot(kind='bar', title=f"Numerikus
{stat_name.capitalize()}")
                plt.ylabel(stat_name.capitalize())
                plt.xticks(rotation=45)
                plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "Nincs numerikus
statisztikai adat a megjelenítéshez.")

        # Visualize categorical statistics
        if 'categorical' in analysis_results and 'value_counts' in
analysis_results['categorical']:
            value_counts_dict =
analysis_results['categorical']['value_counts']
            for col, counts in value_counts_dict.items():
                if counts.nunique() < 20:
                    plt.figure(figsize=(10,6))
                    counts.plot(kind='bar', title=f"Gyakoriság - {col}")
                    plt.xlabel(col)
                    plt.ylabel("Előfordulások száma")
                    plt.xticks(rotation=45)
                    plt.show()
                else:
                    megjelenit_ablak("Figyelmeztetés", f"Túl sok
kategória a diagramhoz: {col}")
        else:
            megjelenit_ablak("Figyelmeztetés", "Nincs kategorikus
statisztikai adat a megjelenítéshez.")

    # Generate combined visualizations
    selected_combined_indices = combined_columns_listbox.curselection()
    selected_combined_columns = [combined_columns_listbox.get(i) for i in
selected_combined_indices]
    combined_data = adatok[selected_combined_columns]
    if not selected_combined_columns:
        megjelenit_ablak("Figyelmeztetés", "Nem választott ki kombinált 
oszlopokat.")
        return

    if combined_visualization_types["Szórásdiagram (Scatter 
Plot)"].get():
        numeric_columns = 
combined_data.select_dtypes(include=["number"]).columns
        if len(numeric_columns) >= 2:
            x_col = numeric_columns[0]
            y_col = numeric_columns[1]
            plt.figure(figsize=(10,6))
            sns.scatterplot(x=combined_data[x_col], 
y=combined_data[y_col])
            plt.title(f"Szórásdiagram: {x_col} vs {y_col}")
            plt.xlabel(x_col)
            plt.ylabel(y_col)
            plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "A szórásdiagramhoz 
legalább két numerikus oszlop szükséges.")

    if combined_visualization_types["Páros Diagram (Pair Plot)"].get():
        numeric_data = combined_data.select_dtypes(include=["number"])
        if len(numeric_data.columns) >= 2:
            sns.pairplot(numeric_data)
            plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "A páros diagramhoz 
legalább két numerikus oszlop szükséges.")

    if combined_visualization_types["Hőtérkép (Heatmap)"].get():
        numeric_data = combined_data.select_dtypes(include=["number"])
        if len(numeric_data.columns) >= 2:
            plt.figure(figsize=(10,8))
            sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm')
            plt.title("Korrelációs mátrix hőtérképe")
            plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "A hőtérképhez legalább 
két numerikus oszlop szükséges.")

    if combined_visualization_types["Boxplot Kategória szerint"].get():
        categorical_columns = 
combined_data.select_dtypes(include=["object", "category"]).columns
        numeric_columns = 
combined_data.select_dtypes(include=["number"]).columns
        if not categorical_columns.empty and not numeric_columns.empty:
            for num_col in numeric_columns:
                for cat_col in categorical_columns:
                    plt.figure(figsize=(10,6))
                    sns.boxplot(x=combined_data[cat_col], 
y=combined_data[num_col])
                    plt.title(f"Boxplot: {num_col} Kategória szerint 
{cat_col}")
                    plt.xlabel(cat_col)
                    plt.ylabel(num_col)
                    plt.xticks(rotation=45)
                    plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "A boxplot kategória
szerint diagramhoz legalább egy numerikus és egy kategorikus oszlop
szükséges.")

    if combined_visualization_types["Csoportosított
Oszlopdiagram"].get():
        categorical_columns =
combined_data.select_dtypes(include=["object", "category"]).columns
        numeric_columns =
combined_data.select_dtypes(include=["number"]).columns
        if not categorical_columns.empty and not numeric_columns.empty:
            for cat_col in categorical_columns:
                for num_col in numeric_columns:
                    grouped_data =
combined_data.groupby(cat_col)[num_col].mean().reset_index()
                    plt.figure(figsize=(10,6))
                    sns.barplot(x=cat_col, y=num_col, data=grouped_data)
                    plt.title(f"Csoportosított Oszlopdiagram: {num_col}
átlagai {cat_col} szerint")
                    plt.xlabel(cat_col)
                    plt.ylabel(f"{num_col} Átlag")
                    plt.xticks(rotation=45)
                    plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "A csoportosított
oszlopdiagramhoz legalább egy numerikus és egy kategorikus oszlop
szükséges.")

    # New visualization for multiple categorical variables
    if combined_visualization_types["Két Kategorikus Változó
Diagramja"].get():
        categorical_columns =
combined_data.select_dtypes(include=["object", "category"]).columns
        if len(categorical_columns) >= 2:
            # Select the first two categorical columns
            cat_col1 = categorical_columns[0]
            cat_col2 = categorical_columns[1]

            # Create a cross-tabulation (contingency table)
            cross_tab = pd.crosstab(combined_data[cat_col1],
combined_data[cat_col2])

            # Heatmap for categorical variables
            plt.figure(figsize=(10, 8))
            sns.heatmap(cross_tab, annot=True, fmt="d", cmap='YlGnBu')
            plt.title(f"Kategorikus Változók Hőtérképe: {cat_col1} vs
{cat_col2}")
            plt.xlabel(cat_col2)
            plt.ylabel(cat_col1)
            plt.show()

            # Stacked Bar Chart
            cross_tab.plot(kind='bar', stacked=True, figsize=(10, 6))
            plt.title(f"Halmozott Oszlopdiagram: {cat_col1} és
{cat_col2}")
```

```
            plt.xlabel(cat_col1)
            plt.ylabel("Előfordulások száma")
            plt.xticks(rotation=45)
            plt.legend(title=cat_col2)
            plt.show()

            # Grouped Bar Chart
            cross_tab.plot(kind='bar', stacked=False, figsize=(10, 6))
            plt.title(f"Csoportosított Oszlopdiagram: {cat_col1} és 
{cat_col2}")
            plt.xlabel(cat_col1)
            plt.ylabel("Előfordulások száma")
            plt.xticks(rotation=45)
            plt.legend(title=cat_col2)
            plt.show()
        else:
            megjelenit_ablak("Figyelmeztetés", "A diagramhoz legalább két 
kategorikus oszlop szükséges.")

    def stacked_bar_chart():
        if data_to_visualize.empty:
            megjelenit_ablak("Figyelmeztetés", "Nincs adat a 
diagramhoz.")
            return

        aggregated_data = 
data_to_visualize.groupby(selected_columns[1:]).size().reset_index(name='
Count')
        plt.figure(figsize=(12,8))
        sns.barplot(data=aggregated_data, x=selected_columns[-1], 
y='Count', hue=selected_columns[0])
        plt.title("Halmozott sávdiagram")
        plt.xticks(rotation=45)
        plt.legend(title=selected_columns[0])
        plt.show()

    def optimized_heatmap():
        if data_to_visualize.empty:
            megjelenit_ablak("Figyelmeztetés", "Nincs adat a 
hőtérképhez.")
            return

        pivot_table = 
data_to_visualize.pivot_table(index=selected_columns[0], 
columns=selected_columns[1], aggfunc='size', fill_value=0)
        plt.figure(figsize=(12,8))
        sns.heatmap(pivot_table, cmap='YlGnBu', annot=False, 
linewidths=0.5)
        plt.title("Optimalizált Hőtérkép")
        plt.show()

# Main program (unchanged)
if __name__ == "__main__":
    root = Tk()
    root.title("Adat Analízis és Vizualizáció")
    root.geometry("200x100")  # Set size for the root window

    # Start button to initiate data loading and analysis options
    Button(
        root,
        text="Adatok Beolvasása",
        command=lambda: [
            adatok := adat_beolvasas(),
            select_analysis_options(adatok) if adatok is not None else
None
        ],
    ).pack(pady=20)

    root.mainloop()  # Single mainloop
```