```
# kaggle CSV - mentalhealth_dataset.csv
import csv
import tkinter as tk

def load_and_process_data(file_path):
    """
    Load and process the cleaned dataset from the CSV file.

    Args:
        file_path (str): Path to the cleaned CSV file.

    Returns:
        dict: Processed data including average CGPA by stress level,
              depression counts by age, and anxiety counts by age.
    """
    cgpa_by_stress = {}
    depression_by_age = {}
    anxiety_by_age = {}
    cgpa_by_gender = {"Male": [], "Female": []}

    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            process_cgpa(cgpa_by_stress, row)
            process_depression(depression_by_age, row)
            process_anxiety(anxiety_by_age, row)
            process_cgpa_by_gender(cgpa_by_gender, row)

    avg_cgpa_by_stress = calculate_average_cgpa(cgpa_by_stress)

    return {
        'avg_cgpa_by_stress': avg_cgpa_by_stress,
        'depression_by_age': depression_by_age,
        'anxiety_by_age': anxiety_by_age,
        'cgpa_by_gender': cgpa_by_gender
    }

def process_cgpa(cgpa_by_stress, row):
    """
    Process CGPA data for each stress level.

    Args:
        cgpa_by_stress (dict): Dictionary to store CGPA data by stress level.
        row (dict): A row of data from the CSV file.
    """
    stress_level = int(row['StudyStressLevel'])
    cgpa = float(row['CGPA'])
    if stress_level not in cgpa_by_stress:
        cgpa_by_stress[stress_level] = []
    cgpa_by_stress[stress_level].append(cgpa)

def process_depression(depression_by_age, row):
    """
    Process depression data for each age.

    Args:
        depression_by_age (dict): Dictionary to store depression counts by age.
        row (dict): A row of data from the CSV file.
    """
```

```
    age = int(row['Age'])
    depression = int(row['Depression'])
    if age not in depression_by_age:
        depression_by_age[age] = 0
    depression_by_age[age] += depression

def process_anxiety(anxiety_by_age, row):
    """
    Process anxiety data for each age.

    Args:
        anxiety_by_age (dict): Dictionary to store anxiety counts by age.
        row (dict): A row of data from the CSV file.
    """
    age = int(row['Age'])
    anxiety = int(row['Anxiety'])
    if age not in anxiety_by_age:
        anxiety_by_age[age] = 0
    anxiety_by_age[age] += anxiety

def process_cgpa_by_gender(cgpa_by_gender, row):
    """
    Process CGPA data for each gender.

    Args:
        cgpa_by_gender (dict): Dictionary to store CGPA data by gender.
        row (dict): A row of data from the CSV file.
    """
    gender = row['Gender']
    cgpa = float(row['CGPA'])
    cgpa_by_gender[gender].append(cgpa)

def calculate_average_cgpa(cgpa_by_stress):
    """
    Calculate the average CGPA for each stress level.

    Args:
        cgpa_by_stress (dict): Dictionary containing CGPA data by stress level.

    Returns:
        dict: Average CGPA for each stress level.
    """
    return {
        stress: sum(cgpas) / len(cgpas)
        for stress, cgpas in cgpa_by_stress.items()
    }

class MentalHealthVisualization:
    """
    A class to create visualizations for mental health data using Tkinter.
    """

    def __init__(self, master, data):
        """
        Initialize the MentalHealthVisualization object.

        Args:
            master (tk.Tk): The root Tkinter window.
            data (dict): Processed mental health data.
```

```
        """
        self.master = master
        self.master.title("Student Mental Health Visualization")
        self.data = data

        self.canvas = tk.Canvas(master, width=800, height=600)
        self.canvas.pack()

        self.draw_visualization()

    def draw_visualization(self):
        """
        Draw the complete visualization including CGPA vs Stress Level,
        Depression/Anxiety by Age charts, and CGPA by Gender.
        """
        self.draw_cgpa_stress()
        self.draw_mental_health_age()
        self.draw_cgpa_distribution()

    def draw_cgpa_stress(self):
        """
        Draw the bar chart for average CGPA vs Study Stress Level with a more
sensible scale.
        """
        avg_cgpa_by_stress = self.data['avg_cgpa_by_stress']

        self.canvas.create_text(200, 30, text="Average CGPA vs Study Stress Level",
font=("Arial", 14, "bold"))

        y_base = 250
        y_top = 50
        y_range = y_base - y_top

        for stress, cgpa in avg_cgpa_by_stress.items():
            x = 100 + stress * 100
            y = y_base - ((cgpa - 2) / 2) * y_range
            self.canvas.create_rectangle(x-20, y_base, x+20, y, fill="blue")
            self.canvas.create_text(x, y_base + 20, text=f"Level {stress}")
            self.canvas.create_text(x, y-15, text=f"{cgpa:.2f}")

        self.canvas.create_line(50, y_base, 550, y_base)
        self.canvas.create_text(300, y_base + 40, text="Study Stress Level")
        self.canvas.create_line(50, y_base, 50, y_top)
        self.canvas.create_text(30, 150, text="CGPA", angle=90)

        for i in range(5):
            y = y_base - (i / 4) * y_range
            self.canvas.create_line(45, y, 55, y)
            self.canvas.create_text(35, y, text=f"{i/2 + 2:.1f}")

    def draw_mental_health_age(self):
        """
        Draw the grouped bar chart for Depression and Anxiety counts by Age with
clearer labels.
        """
        depression_by_age = self.data['depression_by_age']
        anxiety_by_age = self.data['anxiety_by_age']

        if not depression_by_age or not anxiety_by_age:
```

```
            self.canvas.create_text(600, 300, text="No data available for
Depression and Anxiety by Age", font=("Arial", 12))
            return

        max_value = max(max(depression_by_age.values(), default=0),
max(anxiety_by_age.values(), default=0))
        if max_value == 0:
            self.canvas.create_text(600, 300, text="All values are zero for
Depression and Anxiety by Age", font=("Arial", 12))
            return

        self.canvas.create_text(600, 30, text="Depression and Anxiety by Age",
font=("Arial", 14, "bold"))

        y_base = 550
        y_top = 350
        y_range = y_base - y_top
        bar_width = 20

        for age in range(18, 26):
            x = 400 + (age - 18) * 50
            dep_y = y_base - (depression_by_age.get(age, 0) / max_value) * y_range
            anx_y = y_base - (anxiety_by_age.get(age, 0) / max_value) * y_range

            self.canvas.create_rectangle(x-bar_width, y_base, x, dep_y, fill="red")
            self.canvas.create_rectangle(x+1, y_base, x+bar_width+1, anx_y,
fill="green")
            self.canvas.create_text(x, y_base + 20, text=str(age))

        self.canvas.create_line(350, y_base, 750, y_base)
        self.canvas.create_text(550, y_base + 40, text="Age")
        self.canvas.create_line(350, y_base, 350, y_top)
        self.canvas.create_text(330, 450, text="Count", angle=90)

        for i in range(6):
            y = y_base - (i / 5) * y_range
            self.canvas.create_line(345, y, 355, y)
            self.canvas.create_text(335, y, text=str(int(i * max_value / 5)))

        self.canvas.create_rectangle(650, 50, 670, 70, fill="red")
        self.canvas.create_text(725, 60, text="Depression Count", anchor="w")
        self.canvas.create_rectangle(650, 80, 670, 100, fill="green")
        self.canvas.create_text(725, 90, text="Anxiety Count", anchor="w")

    def draw_cgpa_distribution(self):
        """
        Draw a bar chart for CGPA distribution by gender.
        """
        cgpa_by_gender = self.data['cgpa_by_gender']
        x = 50
        y_base = 550
        y_top = 350
        y_range = y_base - y_top

        self.canvas.create_text(200, 330, text="Average CGPA by Gender",
font=("Arial", 14, "bold"))

        for gender, cgpas in cgpa_by_gender.items():
            if cgpas:
```

```
                avg_cgpa = sum(cgpas) / len(cgpas)
                y = y_base - ((avg_cgpa - 2) / 2) * y_range
                self.canvas.create_rectangle(x, y_base, x + 40, y, fill="purple")
                self.canvas.create_text(x + 20, y_base + 20, text=gender)
                self.canvas.create_text(x + 20, y - 15, text=f"{avg_cgpa:.2f}")
            else:
                self.canvas.create_rectangle(x, y_base, x + 40, y_base,
fill="gray")
                self.canvas.create_text(x + 20, y_base + 20, text=gender)
                self.canvas.create_text(x + 20, y_base - 10, text="No data")
            x += 60

        self.canvas.create_line(30, y_base, 150, y_base)
        self.canvas.create_line(30, y_base, 30, y_top)
        self.canvas.create_text(20, 450, text="CGPA", angle=90)

        for i in range(5):
            y = y_base - (i / 4) * y_range
            self.canvas.create_line(25, y, 35, y)
            self.canvas.create_text(15, y, text=f"{i/2 + 2:.1f}")

# Finally the execution

file_path = 'mentalhealth_dataset_cleaned.csv'
processed_data = load_and_process_data(file_path)
root = tk.Tk()
app = MentalHealthVisualization(root, processed_data)
root.mainloop()
```