

Lung Cancer Dataset (from Google Drive) I was change some columns for testing.
<https://drive.google.com/file/d/1AmCBX9nJ7DStH3nJzZ4tUCM226Gy6km/view?usp=sharing>
Original Lunge Cancer Dataset (from the Kaggle) <https://www.kaggle.com/datasets/amankumar094/lung-cancer-dataset>

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

# Importing Required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

# Dataset Loading
df = pd.read_csv("/content/drive/MyDrive/Project1/dataset_med.csv")

# df - head()

# Data Types
print('Getting features type ')
print("\n--- Data Types ---")
print(df.dtypes)

Getting features type
--- Data Types ---
id                int64
age              int64
gender           object
country          object
diagnosis_date   object
cancer_stage     object
family_history   object
smoking_status   object
bmi             float64
cholesterol_level int64
hypertension     int64
asthma           int64
cirrhosis        int64
other_cancer     int64
treatment_type   object
end_treatment_date object
survived         int64
dtype: object

# Missing Values
print("\n--- Missing Values ---")
print(df.isnull().sum())

--- Missing Values ---
id                0
age              0
gender           0
country          0
diagnosis_date   0
cancer_stage     3
family_history   0
smoking_status   8
bmi             0
cholesterol_level 0
hypertension     0
asthma           0
cirrhosis        0
other_cancer     0
treatment_type   5
end_treatment_date 0
survived         0
dtype: int64

# Print columns
df.columns

Index(['id', 'age', 'gender', 'country', 'diagnosis_date', 'cancer_stage',
       'family_history', 'smoking_status', 'bmi', 'cholesterol_level',
       'hypertension', 'asthma', 'cirrhosis', 'other_cancer', 'treatment_type',
       'end_treatment_date', 'survived'],
      dtype='object')

# Deleting columns that have 30% missing values
missing_percentage = df.isnull().mean() * 100
threshold = 30

# # Filter columns with more than the threshold percentage of missing values
cols_to_drop = missing_percentage[missing_percentage > threshold].index
df = df.drop(columns=cols_to_drop)
print("Dropped Columns:", list(cols_to_drop))

Dropped Columns: []

# Pre-processing on dataset
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
print("\n👁️ After Label Encoding:")
print(df.head())

👁️ After Label Encoding:
id age gender country diagnosis_date cancer_stage family_history \
0 1 64 Male Sweden 05-04-16 Stage I Yes
1 2 50 Female Netherlands 20-04-23 Stage III Yes
2 3 65 Female Hungary 05-04-23 Stage III Yes
3 4 51 Female Belgium 05-02-16 Stage I No
4 5 37 Male Luxembourg 29-11-23 Stage I No

smoking_status bmi cholesterol_level hypertension asthma cirrhosis \
0 Passive Smoker 29.4 179 0 0 1
1 Passive Smoker 41.2 280 1 1 0
2 Former Smoker 44.0 268 1 1 0
3 Passive Smoker 43.0 241 1 1 0
4 Passive Smoker 19.7 178 0 0 0

other_cancer treatment_type end_treatment_date survived
0 0 Chemotherapy 10-09-17 0
1 0 Surgery 17-06-24 1
2 0 Combined 09-04-24 0
3 0 Chemotherapy 23-04-17 0
4 0 Combined 08-01-25 0

# Fill missing values in numeric columns with mean
for column in df.select_dtypes(include=['float64', 'int64']).columns:
    mean_value = df[column].mean() # Calculate mean
    df[column] = df[column].fillna(mean_value)

# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the most frequent value
    df[column] = df[column].fillna(mode_value)

# Display remaining missing values (should be all zeros)
print("👁️ Missing values after filling:")
print(df.isnull().sum())

👁️ Missing values after filling:
id                0
age              0
gender           0
country          0
diagnosis_date   0
cancer_stage     0
family_history   0
smoking_status   0
bmi             0
cholesterol_level 0
hypertension     0
asthma           0
cirrhosis        0
other_cancer     0
treatment_type   0
end_treatment_date 0
survived         0
dtype: int64
```

Data Visualization

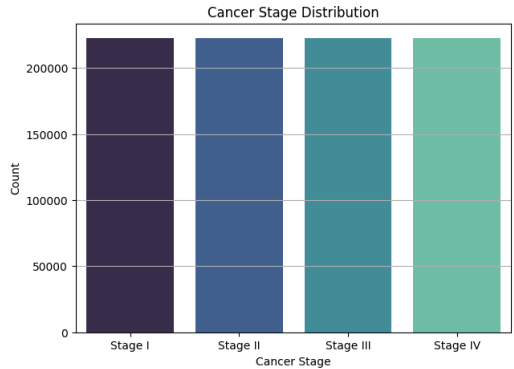
```
# 1. Correlation Heatmap
numeric_df = df.select_dtypes(include=['int64', 'float64'])

plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap - Lung Cancer Dataset')
plt.show()

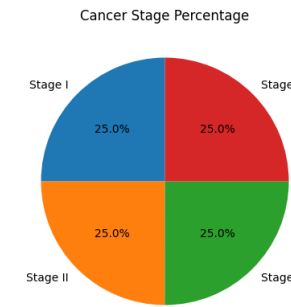
# 2. Classification
df['cancer_stage'].value_counts()

count
cancer_stage
Stage III 222596
Stage IV 222525
Stage I 222516
Stage II 222363
dtype: int64

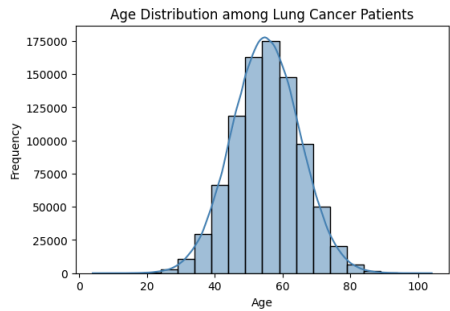
# 3. Cancer Stage Distribution
# 3.1 Bar Chart
if 'cancer_stage' in df.columns:
    plt.figure(figsize=(7, 5))
    stage_counts = df['cancer_stage'].value_counts().sort_index()
    sns.barplot(stage_counts.index, y=stage_counts.values, palette='mako')
    plt.title('Cancer Stage Distribution')
    plt.xlabel('Cancer Stage')
    plt.ylabel('Count')
    plt.grid(axis='y')
    plt.show()
```



```
# 3.2 Pie chart
plt.figure(figsize=(5, 5))
plt.pie(stage_counts, labels=stage_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Cancer Stage Percentage')
plt.show()
```

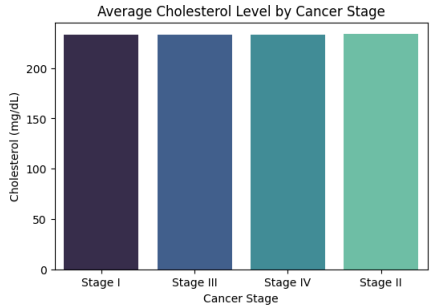


```
# 4. Age Distribution
if 'age' in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df['age'], bins=20, kde=True, color='steelblue')
    plt.title('Age Distribution among Lung Cancer Patients')
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()
```

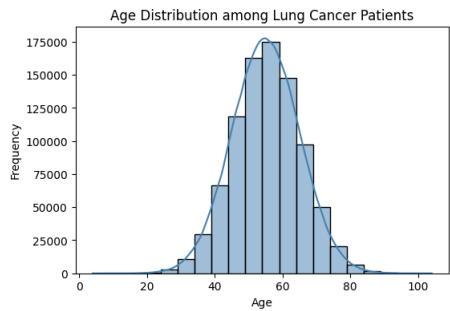


```
# 5. Average Cholesterol Level by Cancer Stage
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(6,4))
sns.barplot(x='cancer_stage', y='cholesterol_level', data=df, estimator='mean', ci=None, palette='mako')
plt.title('Average Cholesterol Level by Cancer Stage')
plt.xlabel('Cancer Stage')
plt.ylabel('Cholesterol (mg/dL)')
plt.show()
```

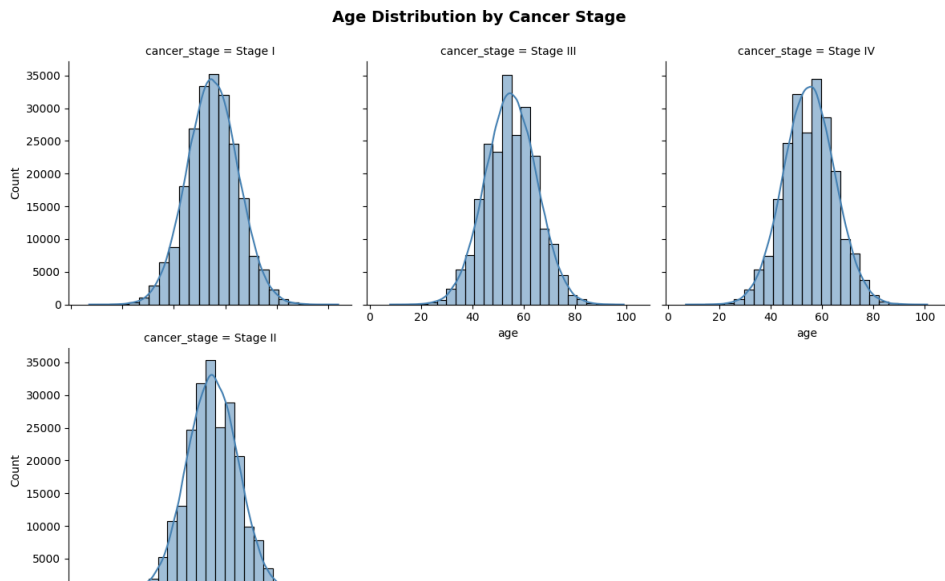


```
# 5.1 Average Age by Cancer Stage
if 'age' in df.columns:
    plt.figure(figsize=(6, 4))
    sns.histplot(df['age'], bins=20, kde=True, color='steelblue')
    plt.title('Age Distribution among Lung Cancer Patients')
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()
```

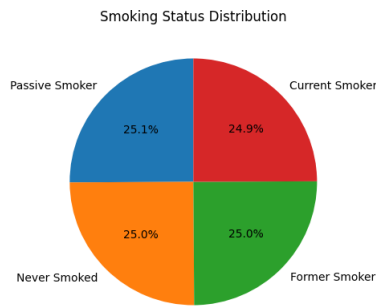


```
# 5.2 Age Distribution by Cancer Stage
g = sns.FacetGrid(df, col='cancer_stage', col_wrap=3, height=4)
g.map(sns.histplot, "age", bins=25, kde=True, color='steelblue')

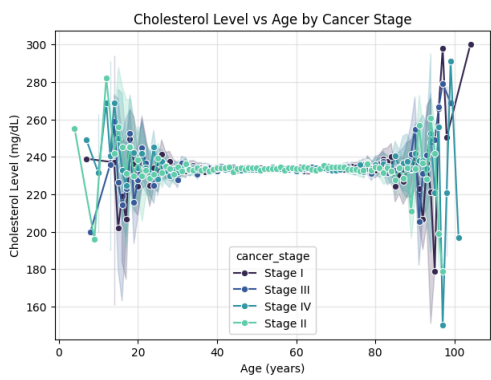
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Age Distribution by Cancer Stage', fontsize=14, fontweight='bold')
plt.show()
```



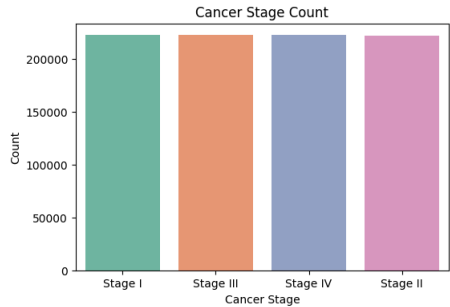
```
# 6. Categorical
# 6.1 distribution of smoking status by Cancer Stage
status_counts = df['smoking_status'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(status_counts, labels=status_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Smoking Status Distribution')
plt.show()
```



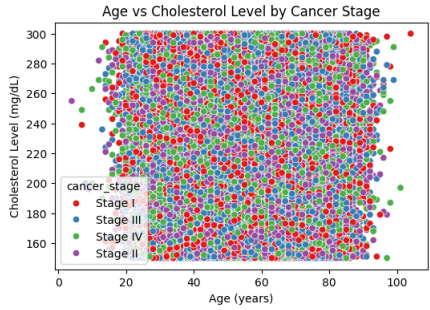
```
# Cholesterol Level (mg/dL)
plt.figure(figsize=(7,5))
sns.lmplot(x='age', y='cholesterol_level', hue='cancer_stage', data=df, marker='o', palette='mako')
plt.title('Cholesterol Level vs Age by Cancer Stage')
plt.xlabel('Age (years)')
plt.ylabel('Cholesterol Level (mg/dL)')
plt.grid(alpha=0.3)
plt.show()
```



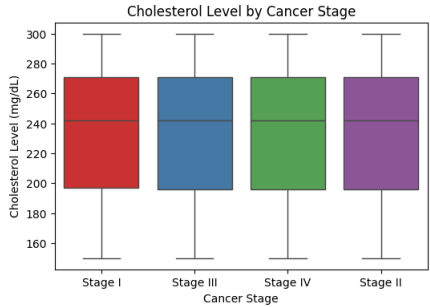
```
# 7. Countplot
# Cancer Stage count
plt.figure(figsize=(6, 4))
sns.countplot(x='cancer_stage', data=df, palette='Set2')
plt.title('Cancer Stage Count')
plt.xlabel('Cancer Stage')
plt.ylabel('Count')
plt.show()
```



```
# 8. scatterplot
# Age vs Cholesterol Level by Cancer Stage
plt.figure(figsize=(6, 4))
sns.scatterplot(x='age', y='cholesterol_level', data=df, hue='cancer_stage', palette='Set1')
plt.title('Age vs Cholesterol Level by Cancer Stage')
plt.xlabel('Age (years)')
plt.ylabel('Cholesterol Level (mg/dl)')
plt.show()
```



```
# Boxplot
plt.figure(figsize=(6, 4))
sns.boxplot(x='cancer_stage', y='cholesterol_level', data=df, palette='Set1')
plt.title('Cholesterol Level by Cancer Stage')
plt.xlabel('Cancer Stage')
plt.ylabel('Cholesterol Level (mg/dl)')
plt.show()
```



```
# 9. colors and data
colors = ['lightcoral', 'skyblue']
labels = ['female', 'male']

# Arranging th Value based on Raci Data
order = df['gender'].value_counts().index

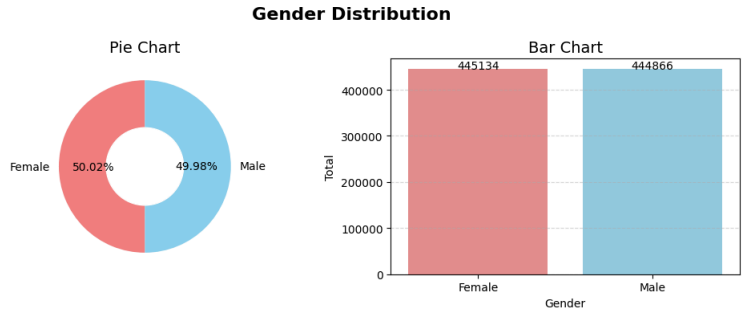
# --- Design ---
plt.figure(figsize=(10, 4))
plt.suptitle('Gender Distribution', fontsize=16, fontweight='bold')

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['gender'].value_counts(), labels=labels, colors=colors, autopct='%2.2F%%', startangle=90)
# 'نسبة ذكور الى اناث 50.02%'
centre = plt.Circle(0, 0, 0.45, color='white')
plt.gca().add_artist(centre)

# --- Bar Chart ---
plt.subplot(1, 2, 2)
plt.title('Bar Chart', fontsize=14)
ax = sns.countplot(x='gender', data=df, palette=colors, order=order)

# عرض القيم فوق الأعمدة
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 1, int(height),
            ha='center', fontsize=18)

plt.xlabel('Gender')
plt.ylabel('Total')
plt.xticks([0, 1], labels)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



```
# Example setup
colors = ['lightpink', 'hotpink', 'deeppink', 'mediumvioletred']
labels = df['treatment_type'].unique() # Automatically get labels from data
order = df['treatment_type'].value_counts().index # Ensure bars follow actual data order

# --- Create Figure ---
plt.figure(figsize=(10, 4))
plt.suptitle('Treatment Type Distribution', fontsize=16, fontweight='bold')

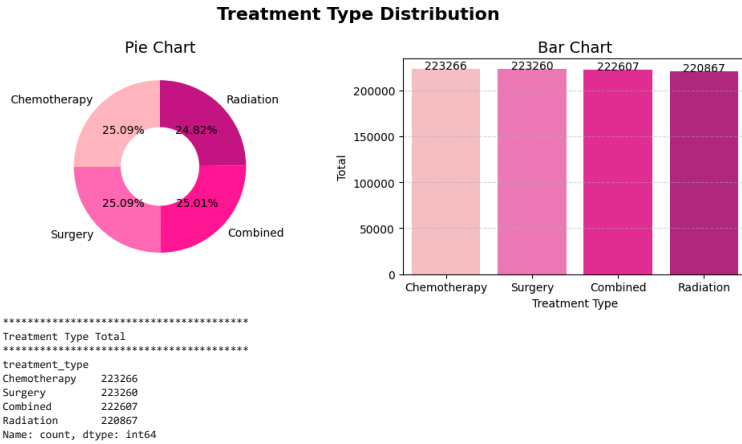
# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['treatment_type'].value_counts(),
        labels=order,
        colors=colors[:len(order)],
        autopct='%2.2F%%',
        startangle=90)
centre = plt.Circle(0, 0, 0.45, color='white') # Donut-style center
plt.gca().add_artist(centre)

# --- Histogram (Count Plot) ---
plt.subplot(1, 2, 2)
plt.title('Bar Chart', fontsize=14)
ax = sns.countplot(x='treatment_type', data=df, palette=colors, order=order)

# Add count labels on top of bars
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2, height + 1, int(height),
            ha='center', fontsize=18)

plt.xlabel('Treatment Type')
plt.ylabel('Total')
plt.xticks([0, 1], labels)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

# --- Display value counts ---
print('--> 40)
print('Treatment Type Total')
print('--> 40)
print(df['treatment_type'].value_counts(dropna=False))
```



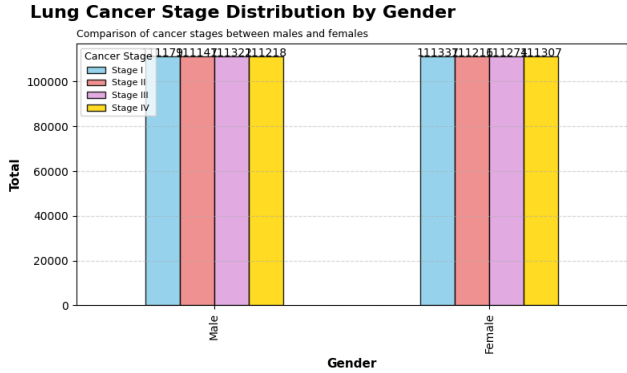
```
# --- Naming ---
labels = df['cancer_stage'].unique() # أسماء مراحل السرطان
label_gender = df['gender'].unique()
label_gender2 = ['Male', 'Female'] # حسب مرحلة يمكن تحديد أن ذكر

# --- Gender Vs Cancer Stage ---
ax = plt.subplots(df['gender'], df['cancer_stage']).plot(
    kind='bar',
    figsize=(8, 5),
    colors='skyblue', 'lightcoral', 'plum', 'gold',
    edgecolor='black',
    alpha=0.85
)

# --- Displaying Numbers above Columns ---
for rect in ax.patches:
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width() / 2, height + 1,
            int(height), ha='center', fontsize=10)

# --- Naming & Subjects ---
plt.suptitle('Lung Cancer Stage Distribution by Gender',
            fontsize=16, fontweight='bold', x=0.85, ha='left')
plt.title('Comparison of cancer stages between males and females',
        fontsize=9, loc='left')
plt.xlabel('gender', fontsize=11, fontweight='bold')
plt.ylabel('Total', fontsize=11, fontweight='bold')
plt.xticks([0, 1], label_gender2)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.grid(axis='x', alpha=0)

plt.legend(title='Cancer Stage', fontsize=8, title_fontsize=9,
        loc='upper left', frameon=True)
plt.tight_layout(rect=[0, 0.84, 1, 1.02])
plt.show()
```



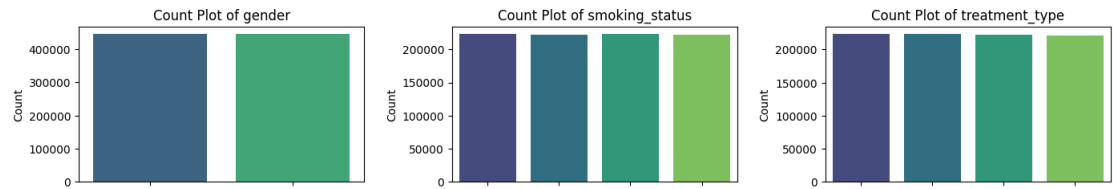
```
# Loop Count Plot
categorical_cols = [
```

```
'gender',
'smoking_status',
'treatment_type',
'treatment_response',
'diet_quality',
'alcohol_use'
]

plt.figure(figsize=(14, 8))

for i, col in enumerate(categorical_cols):
    if col in df.columns:
        plt.subplot(1, 3, i + 1)
        sns.countplot(x=col, data=df, palette='viridis')
        plt.title(f'Count Plot of {col}', fontsize=12)
        plt.xticks(rotation=25)
        plt.xlabel('')
        plt.ylabel('Count')

plt.tight_layout()
plt.show()
```



```
# Columns categories
categorical_cols = [
    'gender',
    'smoking_status',
    'treatment_type',
    'treatment_response',
    'diet_quality'
]

# Number of columns categories
n = len(categorical_cols)
rows, cols = (3, 3)
fig, axes = plt.subplots(rows=rows, ncols=cols, figsize=(18, 12))
axes = axes.flatten()

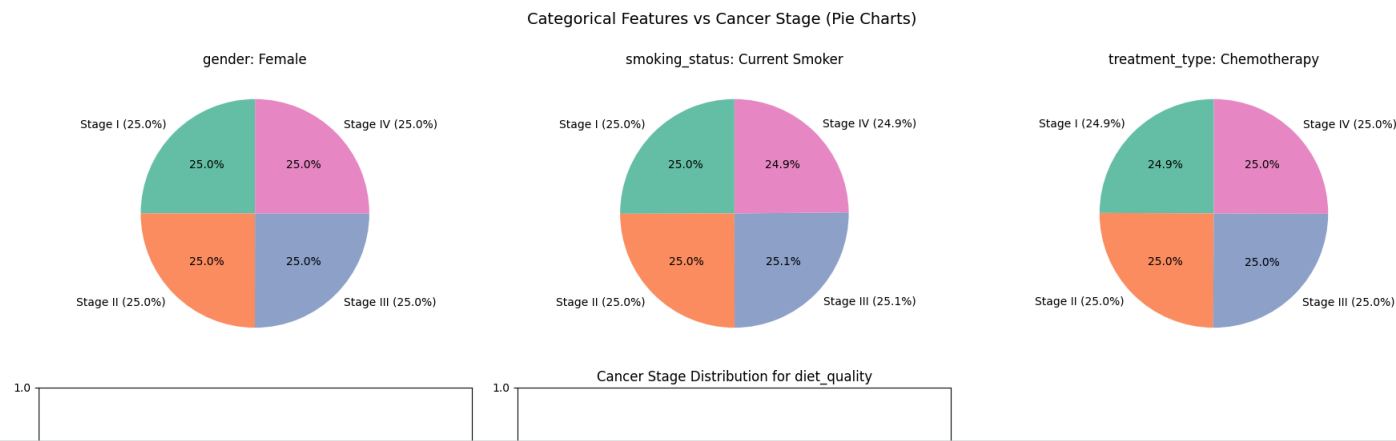
for i, col in enumerate(categorical_cols):
    if col in df.columns:
        # Table creating cross between Cancer Stage & Categories
        ct = pd.crosstab(df[col], df['cancer_stage'], normalize='index')

        # Drawing
        for idx, category in enumerate(ct.index):
            ax = axes[i]
            sizes = ct.loc[category]
            labels = [f'{label} ({sizes[label]*100:.1f}%)' for label in sizes.index]
            ax.pie(
                sizes,
                labels=labels,
                autopct='%1.1f%%',
                startangle=90,
                colors=plt.cm.Set2.colors
            )
            ax.set_title(f'{col}: {category}')
            break

axes[1].set_title(f'Cancer Stage Distribution for {col}')

for j in range(len(categorical_cols), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.suptitle('Categorical Features vs Cancer Stage (Pie Charts)', fontsize=14, y=1.03)
plt.show()
```



```
import math
import seaborn as sns
import matplotlib.pyplot as plt

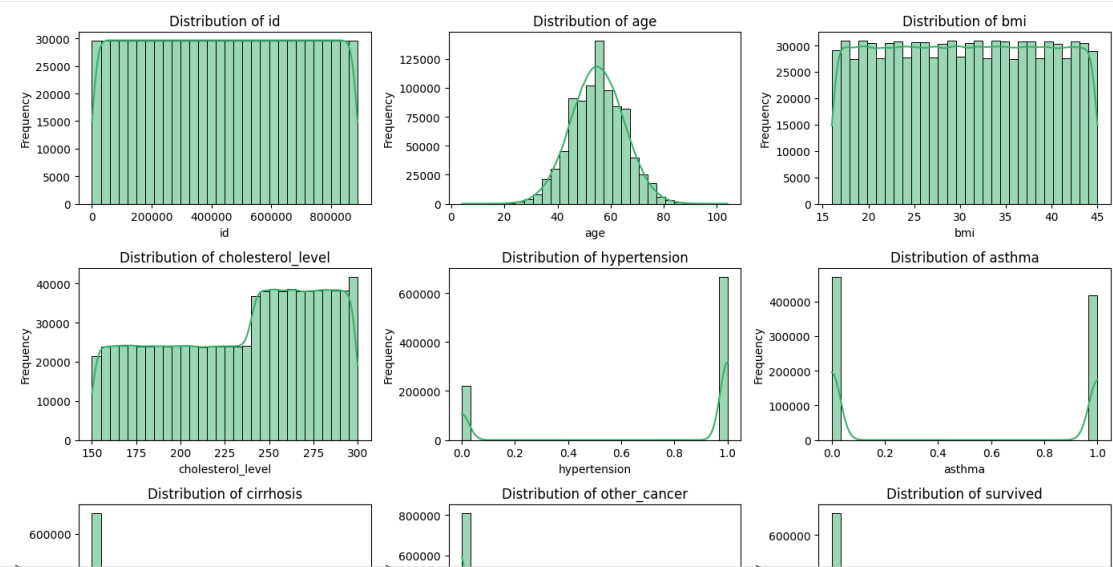
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

n = len(numerical_cols)
cols = 3
rows = math.ceil(n / cols)

plt.figure(figsize=(14, 3 * rows))

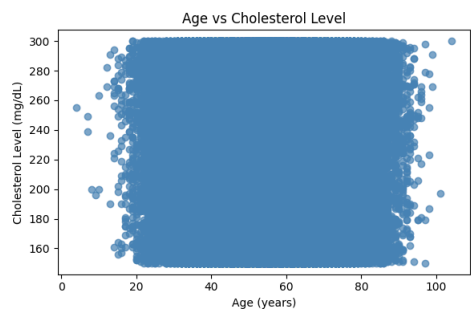
for i, col in enumerate(numerical_cols):
    plt.subplot(rows, cols, i + 1)
    sns.histplot(df[col], kde=True, bins=30, color='mediumseagreen')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



```
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))
plt.scatter(df['age'], df['cholesterol_level'], color='steelblue', alpha=0.7)
plt.title('Age vs Cholesterol Level')
plt.xlabel('Age (years)')
plt.ylabel('Cholesterol Level (mg/dL)')
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(6, 4))
sns.boxplot(x='cancer_stage', y='age', data=df, palette='Set2')
plt.title('Age Distribution by Cancer Stage (Boxplot)')
plt.xlabel('Cancer Stage')
plt.ylabel('Age (years)')
plt.tight_layout()
plt.show()
```

