

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

link dataset <https://www.kaggle.com/datasets/pranavkapratwar/fruit-classification>

```
df = pd.read_csv('/content/drive/MyDrive/Midocean-folder/Project-1/fruit_classification_dataset.csv')
```

```
df.head()
```

	size (cm)	shape	weight (g)	avg_price (₹)	color	taste	fruit_name	
0	25.4	round	3089.2	137.1	green	sweet	watermelon	
1	24.6	round	3283.9	163.8	green	sweet	watermelon	
2	7.8	round	319.0	91.3	green	sweet	custard apple	
3	20.0	oval	1607.0	85.7	orange	sweet	papaya	
4	10.2	long	131.5	37.8	yellow	sweet	banana	

Next steps: [Generate code with df](#) [New interactive sheet](#)

Getting features type

```
df.dtypes
```

```

size (cm)    float64
shape        object
weight (g)   float64
avg_price (₹) float64
color        object
taste        object
fruit_name   object
dtype: object
```

Getting features type

```
df.columns
```

```
Index(['size (cm)', 'shape', 'weight (g)', 'avg_price (₹)', 'color', 'taste',
      'fruit_name'],
      dtype='object')
```

Getting the count of missing values

```
df.isnull().sum()
```

```

0
size (cm) 0
shape 0
weight (g) 0
avg_price (₹) 0
color 0
taste 0
fruit_name 0

dtype: int64

```

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
columns_to_drop = missing_percentage[missing_percentage > threshold].index
# Drop the columns
df = df.drop(columns=columns_to_drop)
# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")

Columns dropped: []
Remaining columns: Index(['size (cm)', 'shape', 'weight (g)', 'avg_price (₹)', 'color', 'taste',
                          'fruit_name'],
                          dtype='object')

```

Pre-processing on dataset

```

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()

```

	size (cm)	shape	weight (g)	avg_price (₹)	color	taste	fruit_name
0	25.4	الإنجليزية		1	2	1	19
1	24.6	العربية		3	2	1	19
2	7.8	شكل		3	2	1	5
3	20.0	شكل, حالة, هيئة		7	3	1	13
4	10.2	اسم		3	7	1	1

Next steps: [Generate code with df](#) [New interactive sheet](#)

Filling missing values based on feature type

```

# Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    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 mode (most frequent value)
    df[column] = df[column].fillna(mode_value)
df.isnull().sum()

```

```
0
size (cm) 0
shape 0
weight (g) 0
avg_price (₹) 0
color 0
taste 0
fruit_name 0

dtype: int64
```

Using Z-Score:

```
# Function to detect outliers in a column using Z-Score
data=df.copy()
def detect_outliers_zscore(column, threshold=3):
    mean = np.mean(column)
    std_dev = np.std(column)
    z_scores = (column - mean) / std_dev
    # Return a boolean series where True indicates an outlier
    return np.abs(z_scores) > threshold
# Apply the Z-Score method to each feature in the DataFrame
for col in data.columns:
    data[f'{col}_Outlier'] = detect_outliers_zscore(data[col])
# Filter the DataFrame to display rows where any outlier flag is True
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]
print(outliers_only)
```

	size (cm)	shape	weight (g)	avg_price (₹)	color	taste	fruit_name \
0	25.4	2	3089.2	137.1	2	1	19
1	24.6	2	3283.9	163.8	2	1	19
43	27.2	2	2818.8	148.4	2	1	19
50	24.8	2	3274.4	148.3	2	1	19
79	24.5	2	2965.6	163.1	2	1	19
...	...	...	...	...	...	...	...
9905	24.1	2	3114.5	144.8	2	1	19
9911	24.4	2	3220.5	156.0	2	1	19
9936	26.6	2	3017.0	148.2	2	1	19
9952	26.3	2	3190.8	136.4	2	1	19
9983	22.5	2	3202.0	156.5	2	1	19

	size (cm)_Outlier	shape_Outlier	weight (g)_Outlier \
0	False	False	True
1	False	False	True
43	False	False	True
50	False	False	True
79	False	False	True
...	...	...	...
9905	False	False	True
9911	False	False	True
9936	False	False	True
9952	False	False	True
9983	False	False	True

	avg_price (₹)_Outlier	color_Outlier	taste_Outlier	fruit_name_Outlier
0	False	False	False	False
1	False	False	False	False
43	False	False	False	False
50	False	False	False	False
79	False	False	False	False
...	...	...	...	...
9905	False	False	False	False
9911	False	False	False	False
9936	False	False	False	False
9952	False	False	False	False
9983	False	False	False	False

[498 rows x 14 columns]

Handling Outliers

```
data=df.copy()
def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with NaN
    return column.where((column >= lower_bound) & (column <=
upper_bound), np.nan)
# Replace outliers with NaN for each feature
for col in data.columns:
    data[col] = replace_outliers_with_nan(data[col])
null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)
```

```
Sum of null values in each column:
size (cm)      804
shape          0
weight (g)     2042
avg_price (₹)   0
color          0
taste          2496
fruit_name     0
dtype: int64
```

We use K-Nearest Neighbors (KNN) Imputation to handle missing values

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2)
data_imputed_knn = pd.DataFrame(imputer.fit_transform(data),
columns=data.columns)
data_imputed_knn.isnull().sum()
```

```
taste
العربية
ذوق
اسم
طعم, ذوق, نكهة
تذوق, ذائق
فعل
خيارات الإضافة
المزيد
taste      0
fruit_name  0
dtype: int64
```

Using the Interquartile Range (IQR) Method

```
data=df.copy()
def remove_outliers_iqr(data):
    # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each feature
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1

    # Define lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filter the dataset to remove rows with outliers in any column
    data_cleaned = data[~((data < lower_bound) | (data >
upper_bound)).any(axis=1)]

    return data_cleaned
cleaned_data = remove_outliers_iqr(data)
print("Original Data:")
print(len(data))
print("\nCleaned Data (without outliers):")
```

```
print(len(cleaned_data))
```

Original Data:
10000

Cleaned Data (without outliers):
5995

Using the Z-Score Method

```
data = df.copy() # Create a copy to avoid modifying the original dataframe

def remove_outliers_zscore(data, threshold=3):

    # Boolean DataFrame indicating if a value is an outlier
    outliers = pd.DataFrame(False, index=df.index, columns=df.columns)

    # Iterate through each numeric column
    for column in data.columns:
        if data[column].dtype in ['float64', 'int64']: # Only apply to numeric columns
            mean = data[column].mean()
            std_dev = data[column].std()
            z_scores = (data[column] - mean) / std_dev

    # Mark True in outliers DataFrame where abs(z_score) > threshold
    outliers[column] = abs(z_scores) > threshold

    # Filter the rows that have any True in the outliers DataFrame (indicating an outlier)
    rows_to_keep = ~outliers.any(axis=1)

    # Return the DataFrame with outlier rows removed
    return data[rows_to_keep]

# Example usage:
cleaned_data = remove_outliers_zscore(data, threshold=3)
print("Original Data:")
print(len(data))
print("\nCleaned Data (without outliers):")
print(len(cleaned_data))
```

Original Data:
10000

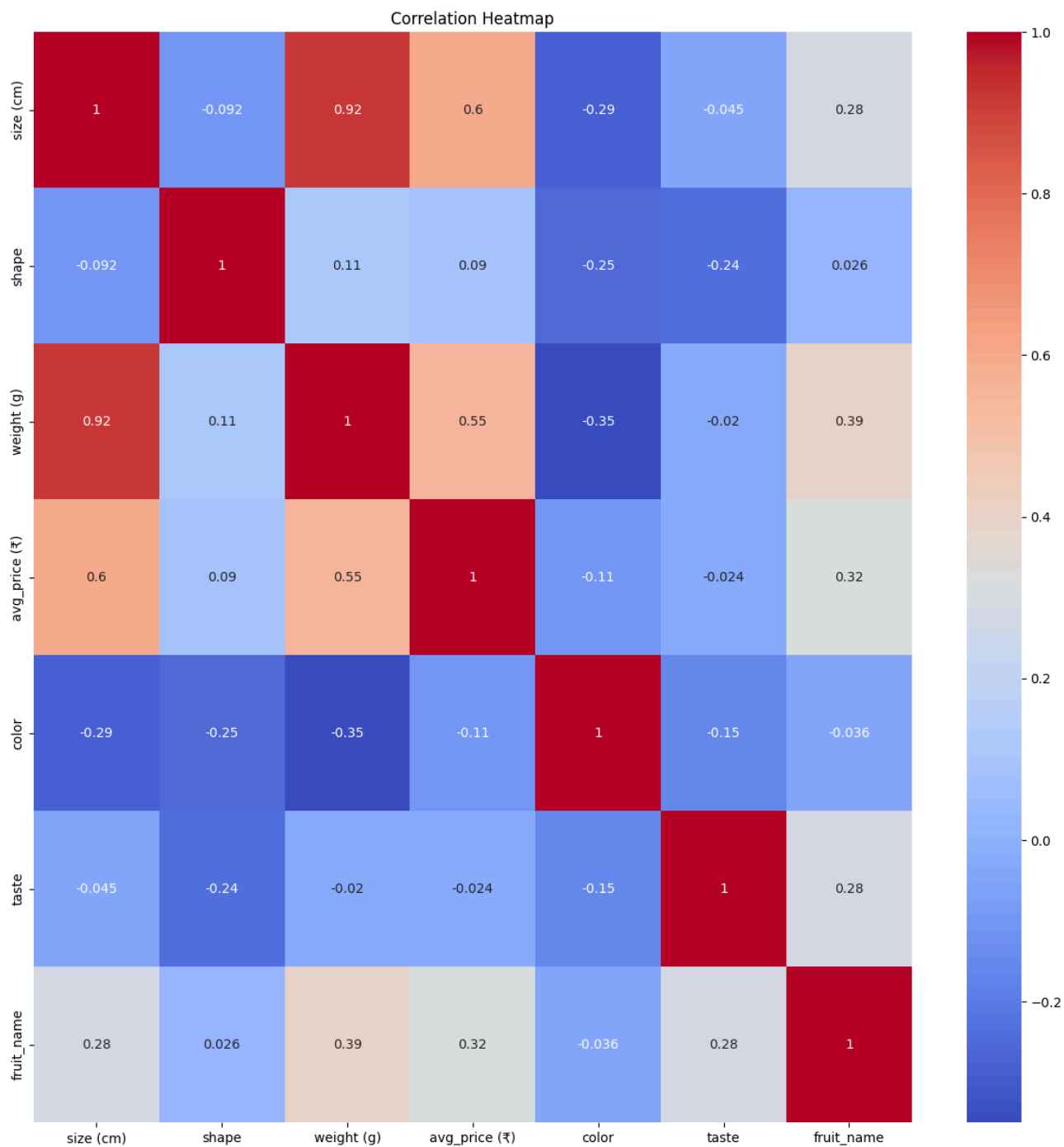
Cleaned Data (without outliers):
9502

Data Visualization:

Correlation heatmap

```
plt.figure(figsize=(15, 15))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')

plt.show()
```



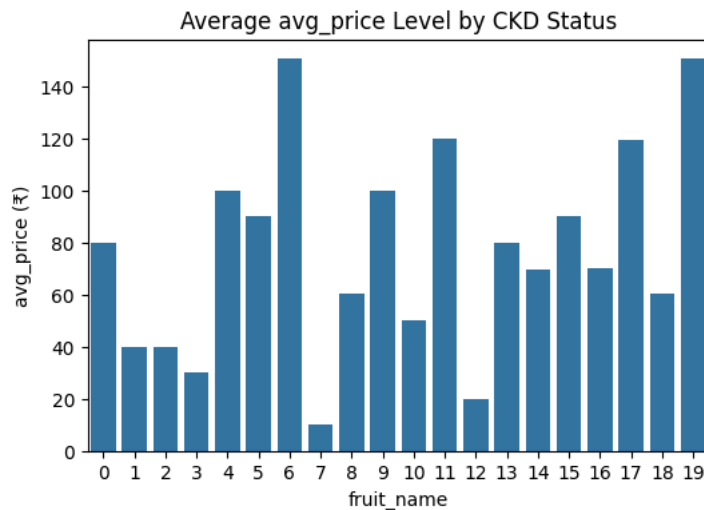
Double-click (or enter) to edit

```
df['fruit_name'].value_counts()
```

dtype: int64

Class	Count
0	530
1	530
2	530
3	510
4	510
5	510
6	500
7	500
8	500
9	500
10	500
11	500
12	500
13	490
14	490
15	490
16	480
17	480
18	480
19	440

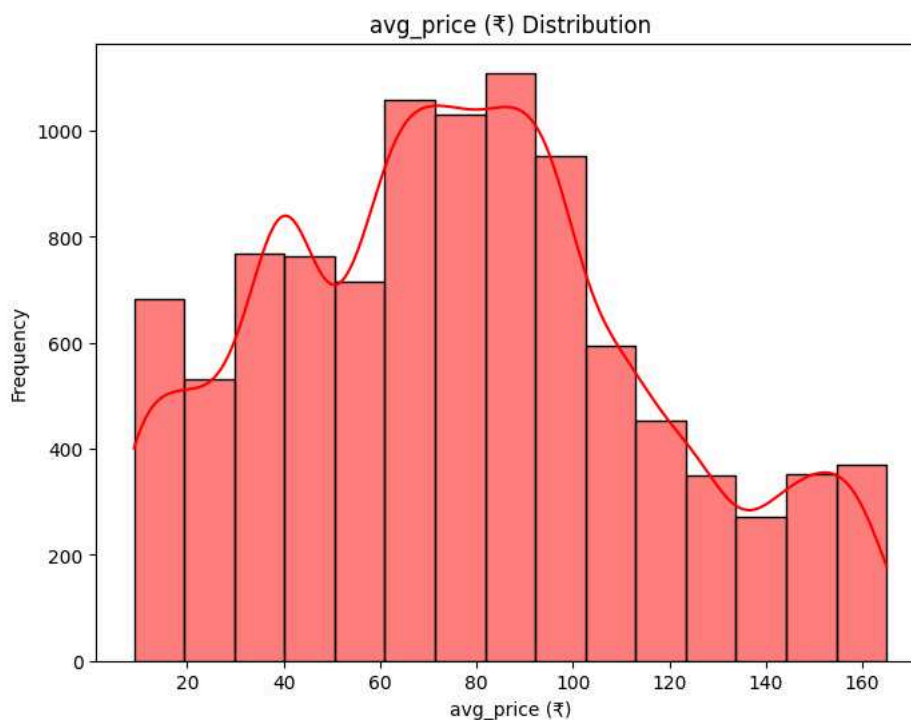
```
# Example 2: Average avg_price by CKD Status
import seaborn as sns
plt.figure(figsize=(6,4))
sns.barplot(x='fruit_name', y='avg_price (₹)', data=df, estimator='mean', ci=None)
plt.title("Average avg_price Level by CKD Status")
plt.xlabel("fruit_name")
plt.ylabel("avg_price (₹)")
plt.show()
```



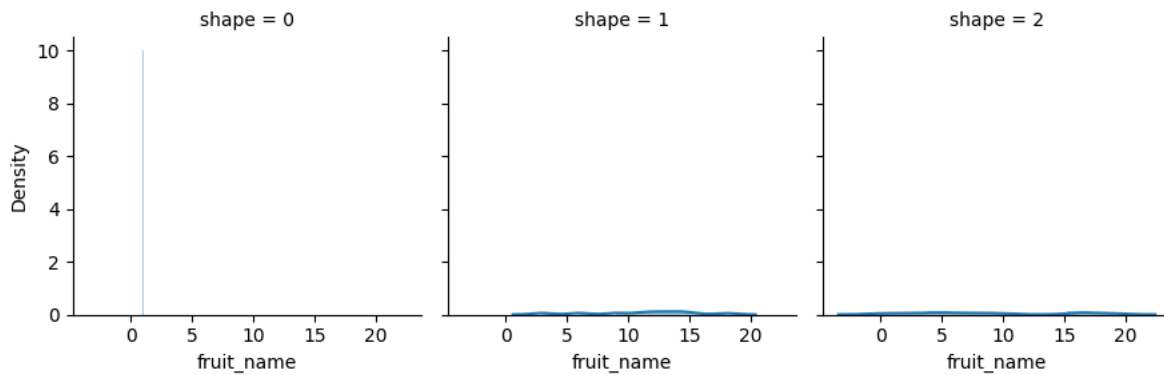
```
cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
```

```
for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')
```

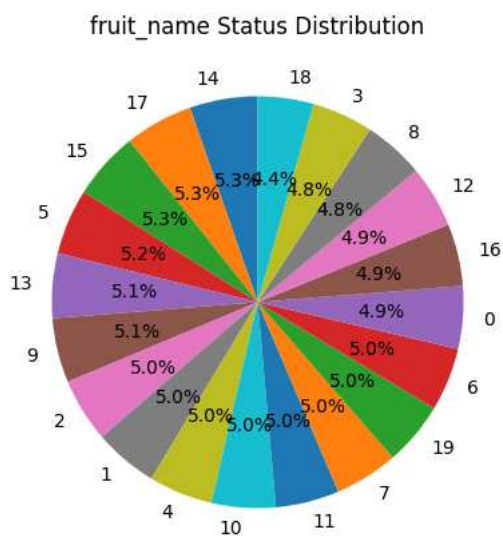
```
# 1. Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['avg_price (₹)'], bins=15, kde=True, color='red')
plt.title('avg_price (₹) Distribution')
plt.xlabel('avg_price (₹)')
plt.ylabel('Frequency')
plt.show()
```



```
g = sns.FacetGrid(df,col="shape")
g.map(sns.distplot,"fruit_name", bins=10)
plt.show()
```

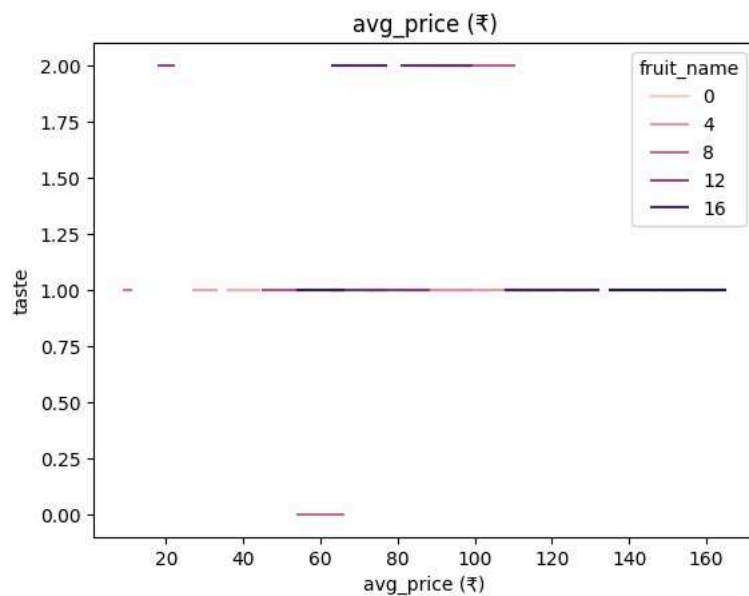


```
appet_counts = df['fruit_name'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%',startangle=90)
plt.title("fruit name Status Distribution")
plt.show()
```

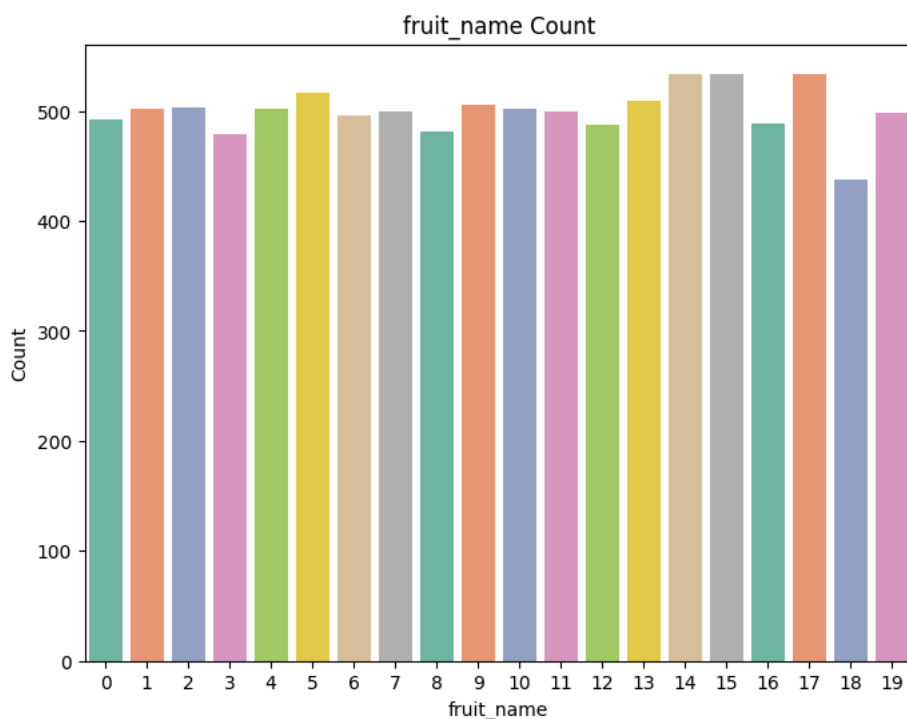


Double-click (or enter) to edit

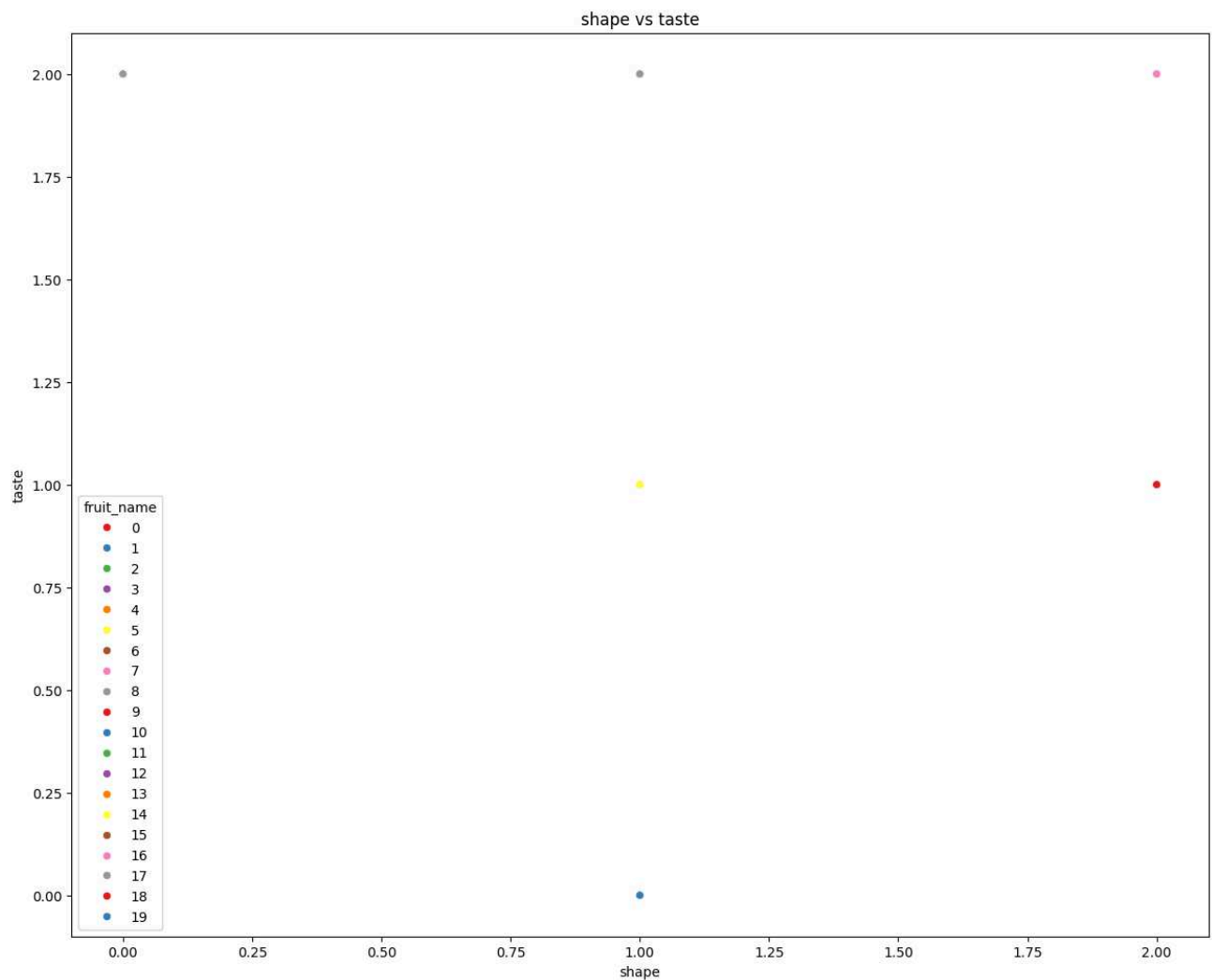
```
sns.lineplot(x='avg_price (₹)', y='taste', hue='fruit_name', data=df)
plt.title("avg_price (₹)")
plt.xlabel("avg_price (₹)")
plt.ylabel("taste")
plt.show()
```



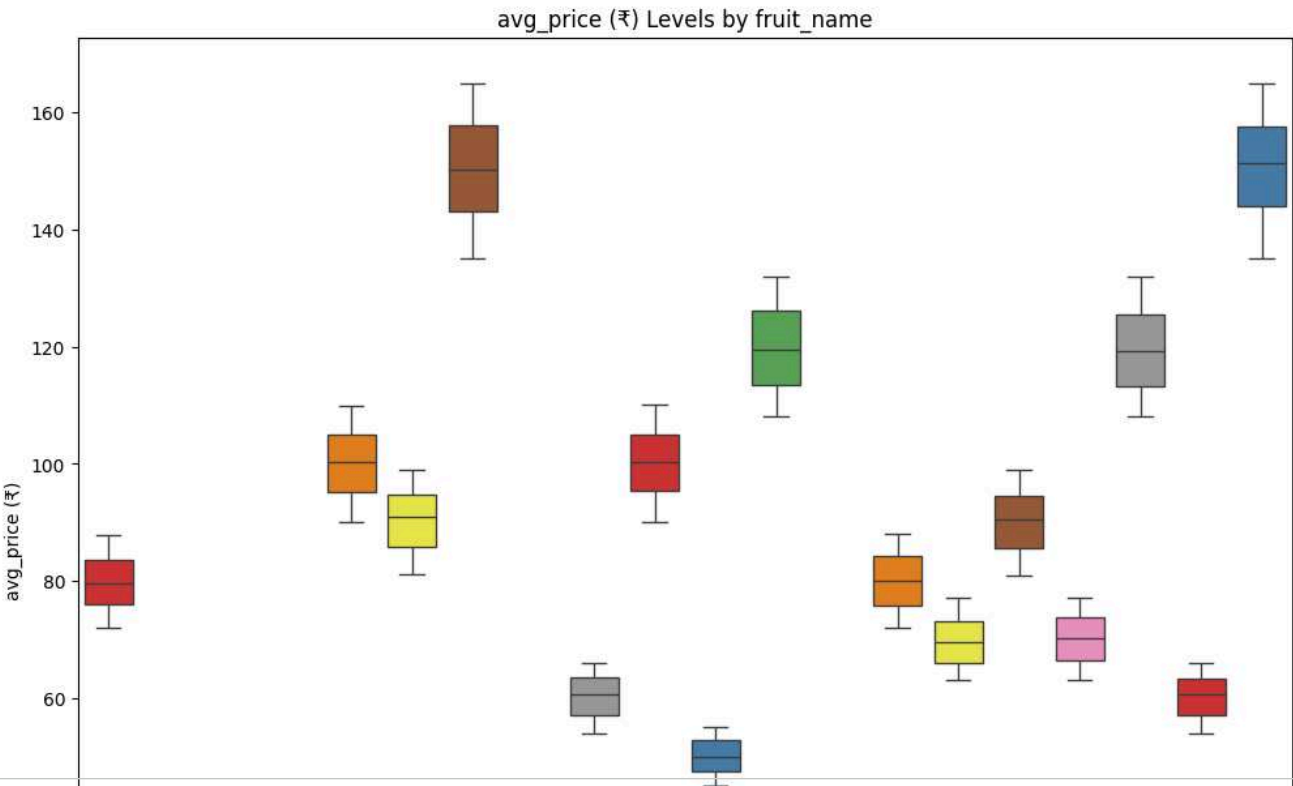
```
# 3. Countplot for Classification
plt.figure(figsize=(8, 6))
sns.countplot(x='fruit_name', data=df, palette='Set2')
plt.title('fruit_name Count')
plt.xlabel('fruit_name')
plt.ylabel('Count')
plt.show()
```



```
plt.figure(figsize=(15, 12))
sns.scatterplot(x='taste', y='shape', data=df, hue='fruit_name', palette='Set1')
plt.title('shape vs taste')
plt.xlabel('shape')
plt.ylabel('taste')
plt.show()
```



```
# Boxplot for Hemoglobin by Target
plt.figure(figsize=(12, 10))
sns.boxplot(x='fruit_name', y='avg_price (₹)', data=df, palette='Set1')
plt.title('avg_price (₹) Levels by fruit_name')
plt.xlabel('fruit_name')
plt.ylabel('avg_price (₹)')
plt.show()
```



```
df1 = pd.read_csv('/content/drive/MyDrive/Midocean-folder/Project-1/eye_cancer_patients.csv')
df1.head()
```

	Patient_ID	Age	Gender	Cancer_Type	Laterality	Date_of_Diagnosis	Stage_at_Diagnosis	Treatment_Type	Surgery_Status	Rad
0	PID00001	58	F	Retinoblastoma	Left	2019-01-25	Stage IV	Radiation	False	
1	PID00002	15	Other	Retinoblastoma	Right	2021-10-21	Stage III	Chemotherapy	True	
2	PID00003	64	M	Retinoblastoma	Bilateral	2021-03-12	Stage IV	Surgery	False	
3	PID00004	33	M	Melanoma	Right	2021-05-18	Stage II	Radiation	True	
4	PID00005	8	Other	Lymphoma	Left	2019-11-24	Stage I	Chemotherapy	False	

eye-cancer-patients <https://www.kaggle.com/datasets/skyshines/eye-cancer-patients>

```
numerical_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
categorical_cols = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca']
df1.describe()
```

	Age	Radiation_Therapy	Chemotherapy	Survival_Time_Months
count	5000.000000	5000.000000	5000.000000	5000.000000
mean	45.008000	35.447600	10.057400	60.716000
std	25.934387	20.565754	6.030968	34.549774
min	1.000000	0.000000	0.000000	1.000000
25%	22.000000	18.000000	5.000000	31.000000
50%	44.000000	35.000000	10.000000	60.000000
75%	67.000000	53.000000	15.000000	90.000000
max	90.000000	70.000000	20.000000	120.000000

```
df1.shape
```

```
(5000, 16)
```

```
df1.isnull().sum()
```

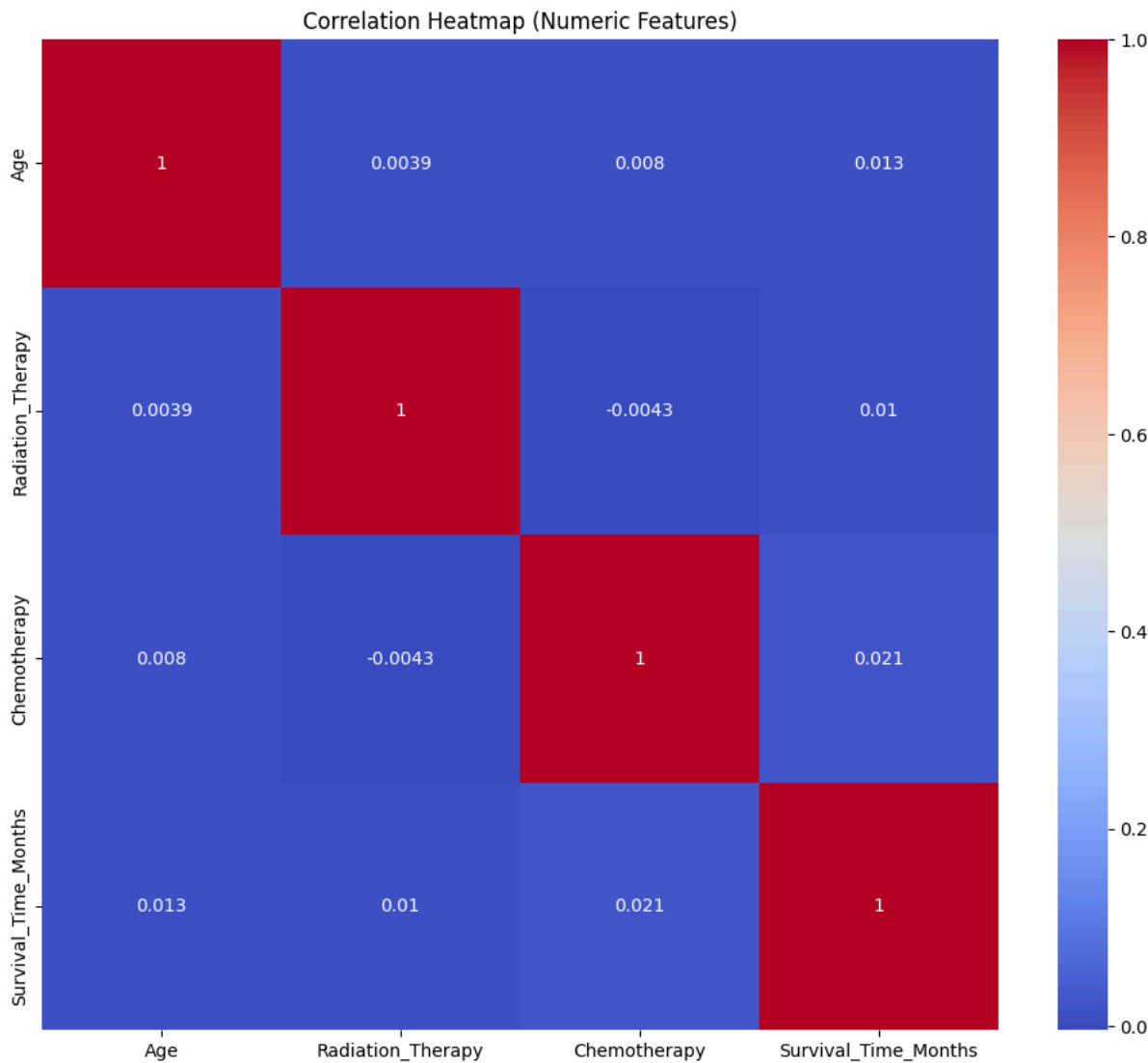
	$\emptyset$
Patient_ID	0
Age	0
Gender	0
Cancer_Type	0
Laterality	0
Date_of_Diagnosis	0
Stage_at_Diagnosis	0
Treatment_Type	0
Surgery_Status	0
Radiation_Therapy	0
Chemotherapy	0
Outcome_Status	0
Survival_Time_Months	0
Genetic_Markers	2497
Family_History	0
Country	0

dtype: int64

```
num_df = df1.select_dtypes(include=['number'])

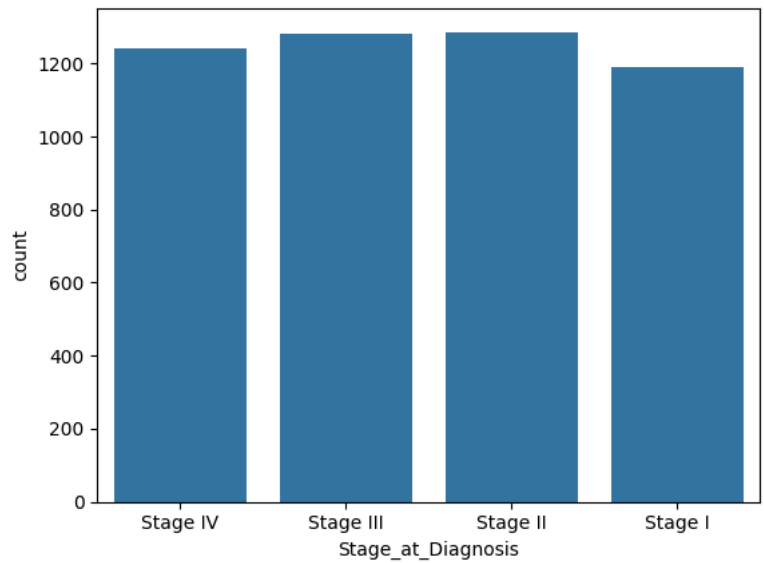
corr = num_df.corr(numeric_only=True)

plt.figure(figsize=(12, 10))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap (Numeric Features)')
plt.show()
```



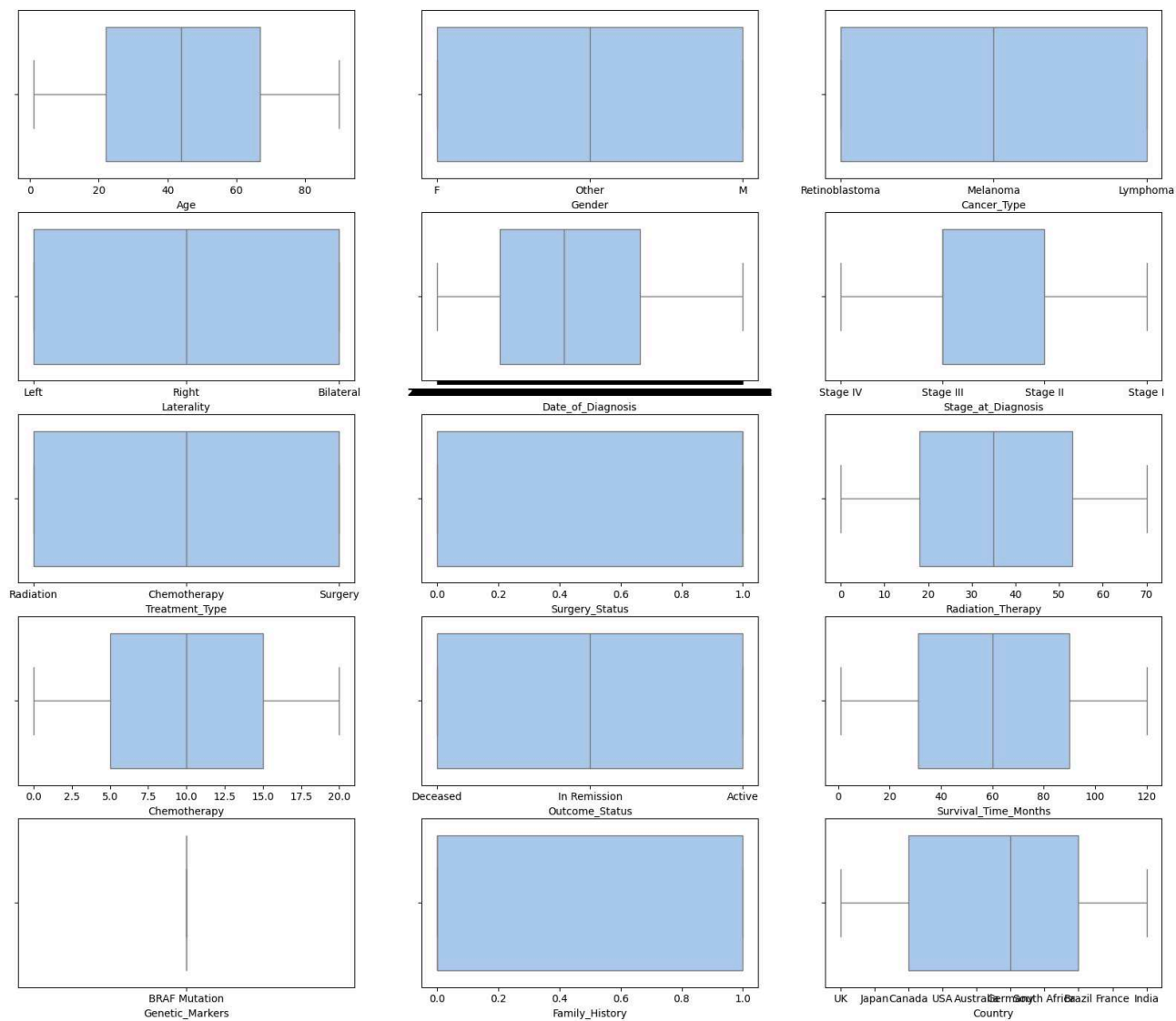
```
sns.countplot(x='Stage_at_Diagnosis',data=df1)
```

<Axes: xlabel='Stage_at_Diagnosis', ylabel='count'>



```
plt.figure(figsize=(20,35))
plotnumber =1
for column in df1.columns[1:]:
    ax = plt.subplot(10,3,plotnumber)
```

```
sns.boxplot(data = df1, x = column, palette='pastel')
plt.xlabel(column)
plotnumber+=1
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# الألوان (يمكنك تعديلها حسب ذوقك)
colors = ['lightcoral', 'skyblue']

# قراءة القيم والفئات الحقيقية من العمود
gender_counts = df1['Gender'].value_counts()
```

```

labels = gender_counts.index

# ترتيب حسب القيم
order = gender_counts.index

# --- إعداد الشكل ---
plt.figure(figsize=(16, 8))
plt.suptitle('Sex (Gender) Distribution', fontsize=16, fontweight='bold')

# --- المخطط الدائري ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(gender_counts, labels=labels, colors=colors[:len(labels)], autopct='%0.2f%%', startangle=90)
centre = plt.Circle((0, 0), 0.45, color='white') # تأثير الدونات
plt.gca().add_artist(centre)

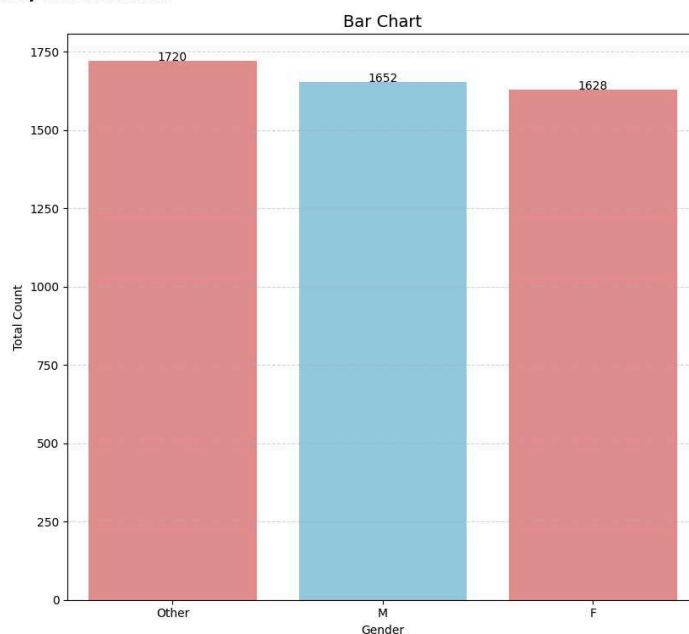
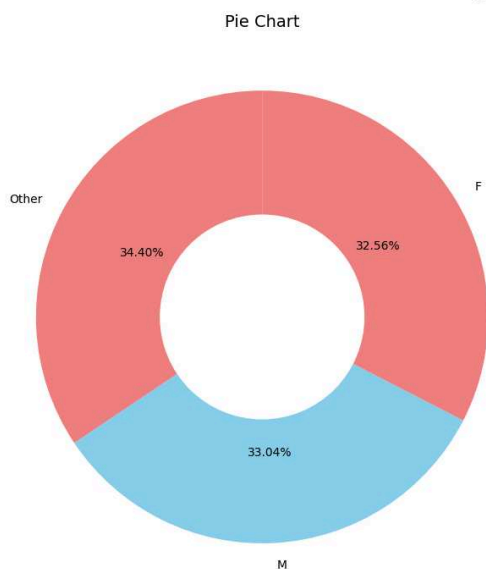
# --- مخطط الأعمدة ---
plt.subplot(1, 2, 2)
plt.title('Bar Chart', fontsize=14)
ax = sns.countplot(x='Gender', data=df1, palette=colors[:len(labels)], order=order)

# إضافة الأرقام فوق الأعمدة
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 0.5, int(height), ha='center', fontsize=10)

plt.xlabel('Gender')
plt.ylabel('Total Count')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

```

Sex (Gender) Distribution



```

import matplotlib.pyplot as plt
import pandas as pd

# اختر العمود الفئوي الذي تريد رسمه
col = 'Cancer_Type' # مثال: 'Gender' أو 'Stage_at_Diagnosis'

```

```

counts = df1[col].value_counts(dropna=False)
labels = counts.index.astype(str)

plt.figure(figsize=(16, 8))
plt.suptitle(f'{col} Distribution', fontsize=16, fontweight='bold')

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(counts, labels=labels, autopct='%.2f%%', startangle=90)
centre = plt.Circle((0, 0), 0.45, color='white') # تأثير الدونات
plt.gca().add_artist(centre)

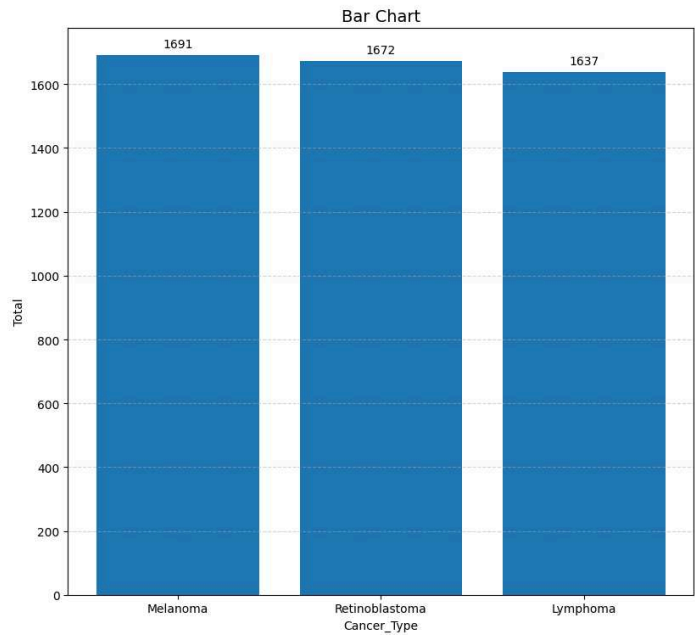
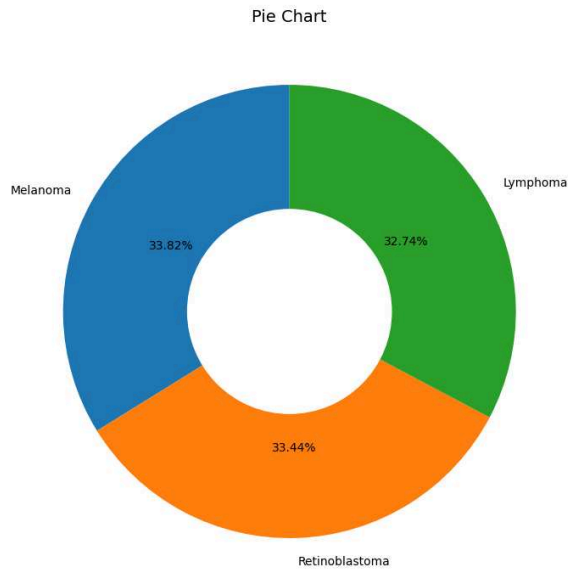
# --- Bar Chart (matplotlib مشاكل الاعتمادات فقط لتفادي) ---
plt.subplot(1, 2, 2)
plt.title('Bar Chart', fontsize=14)
bars = plt.bar(labels, counts.values)
# أرقام فوق الأعمدة
for b in bars:
    h = b.get_height()
    plt.text(b.get_x() + b.get_width()/2, h + max(counts.values)*0.01,
             int(h), ha='center', va='bottom', fontsize=10)

plt.xlabel(col)
plt.ylabel('Total')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()

# لعرض المجاميع نصيًا
print('*' * 30)
print(f'{col} Totals')
print('*' * 30)
print(counts)

```

Cancer_Type Distribution



```
*****
Cancer_Type Totals
*****
Cancer_Type
Melanoma      1691
Retinoblastoma 1672
Lymphoma      1637
Name: count, dtype: int64
```

```

import pandas as pd
import matplotlib.pyplot as plt

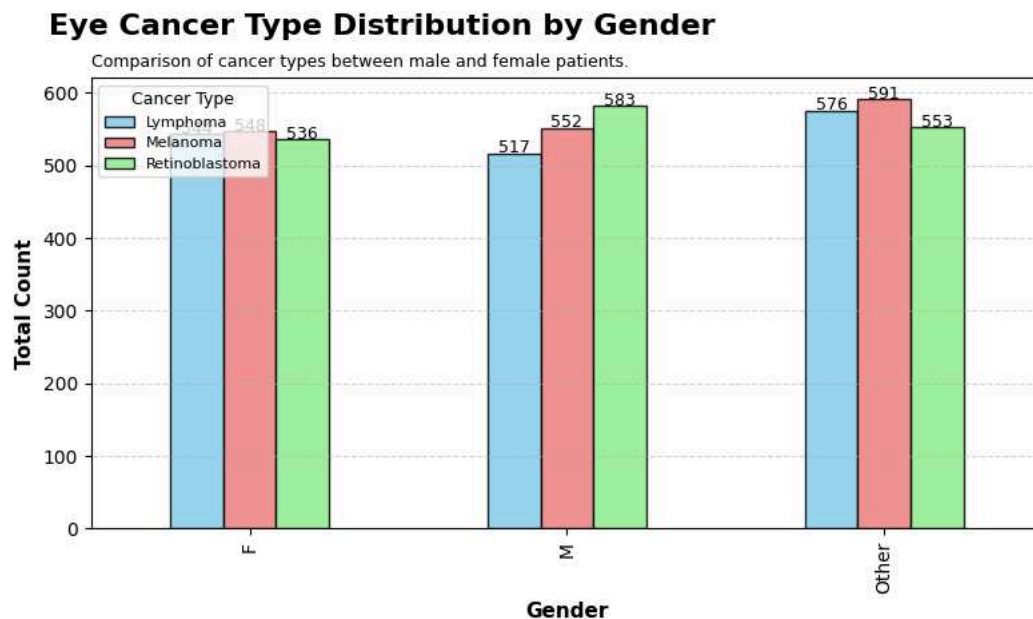
# --- Label Settings ---
labels = df1['Cancer_Type'].unique().tolist() # أسماء أنواع السرطان الموجودة

# --- Create Bar Chart ---
ax = pd.crosstab(df1['Gender'], df1['Cancer_Type']).plot(
    kind='bar',
    figsize=(8, 5),
    color=['skyblue', 'lightcoral', 'lightgreen', 'gold'],
    edgecolor='black',
    alpha=0.85
)

# --- Annotate Bars ---
for rect in ax.patches:
    height = rect.get_height()
    if height > 0:
        ax.text(rect.get_x() + rect.get_width() / 2, height + 0.2,
                int(height), ha='center', fontsize=9)

# --- Titles and Labels ---
plt.suptitle('Eye Cancer Type Distribution by Gender', fontsize=16, fontweight='bold', x=0.05, ha='left')
plt.title("Comparison of cancer types between male and female patients.", fontsize=9, loc='left')
plt.xlabel('Gender', fontsize=11, fontweight='bold')
plt.ylabel('Total Count', fontsize=11, fontweight='bold')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.legend(title='Cancer Type', fontsize=8, title_fontsize=9, loc='upper left', frameon=True)
plt.tight_layout(rect=[0, 0.04, 1, 1.02])
plt.show()

```



```

import matplotlib.pyplot as plt
import seaborn as sns

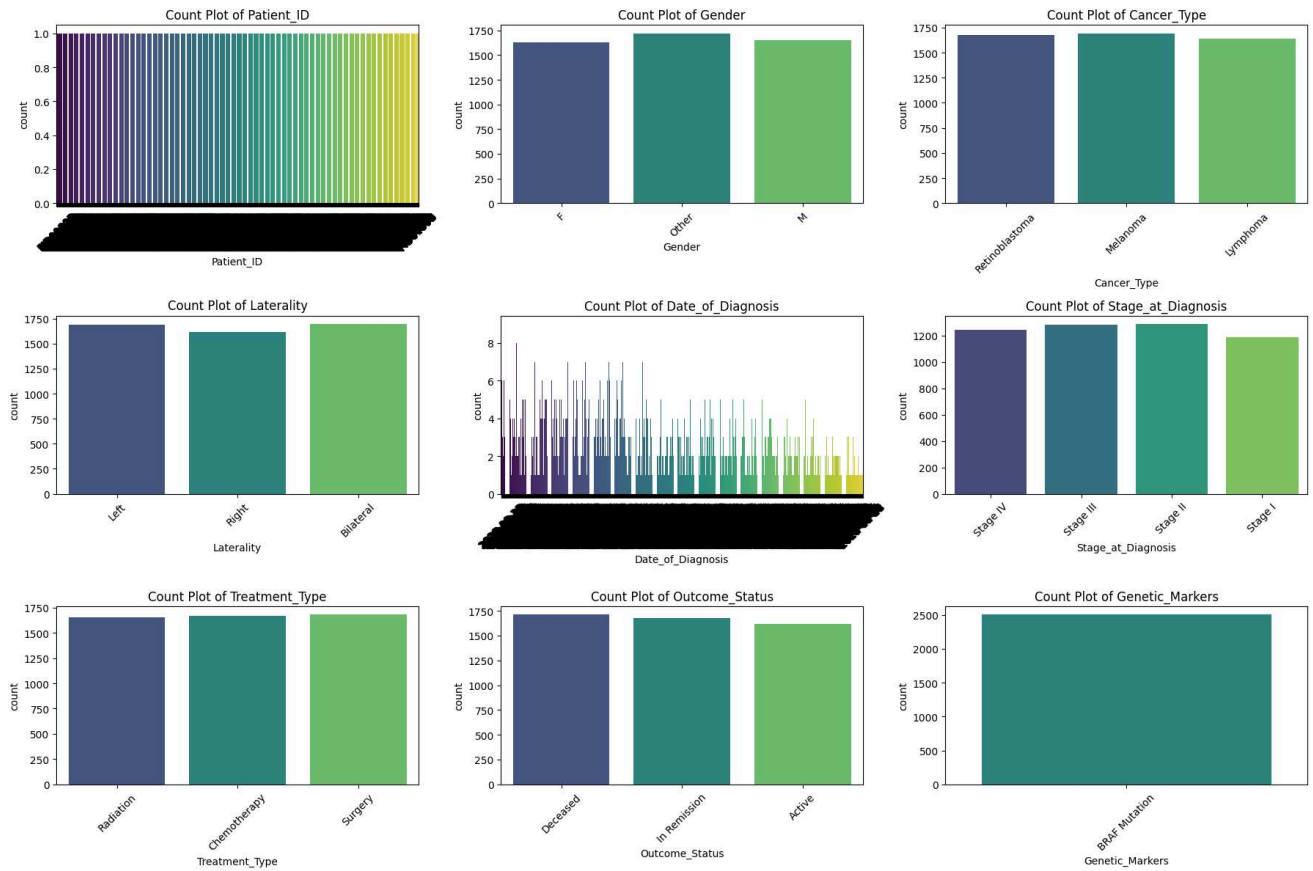
categorical_cols = df1.select_dtypes(include='object').columns.tolist()

plt.figure(figsize=(18, 12))

for i, col in enumerate(categorical_cols[:9], 1): # نرسم أول 9 فقط
    plt.subplot(3, 3, i)
    sns.countplot(x=col, data=df1, palette='viridis')
    plt.title(f'Count Plot of {col}')
    plt.xticks(rotation=45)
    plt.tight_layout()

plt.show()

```



```
import pandas as pd
import matplotlib.pyplot as plt

DF = df1 # بياناتك
target_col = 'Outcome_Status' # العمود الهدف المختار يدوياً

# الأعمدة القوية (باستثناء الهدف)
categorical_cols = DF.select_dtypes(include='object').columns.tolist()
if target_col in categorical_cols:
    categorical_cols.remove(target_col)

# شبكة العرض
n = len(categorical_cols)
nrows, ncols = 3, 3
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(18, 12))
axes = axes.flatten()

for i, col in enumerate(categorical_cols[:len(axes)]):
    ax = axes[i]
    ct = pd.crosstab(DF[col], DF[target_col], normalize='index')

    bottom = None
    for j, t in enumerate(ct.columns):
        vals = ct[t].values
        ax.bar(ct.index.astype(str), vals, bottom=bottom, label=str(t))
        bottom = vals if bottom is None else bottom + vals

    ax.set_title(f'{col} - Distribution by {target_col}')
```

```
ax.set_xticklabels(ct.index.astype(str), rotation=45, ha='right')
ax.set_ylim(0, 1)
ax.set_ylabel('Proportion')
ax.grid(axis='y', linestyle='--', alpha=0.4)

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

# الأسطورة والعنوان
handles, labels = axes[0].get_legend_handles_labels()
fig.legend(handles, labels, title=target_col, loc='upper center', ncol=len(labels), frameon=True)
plt.suptitle(f'Categorical Features vs {target_col} - 100% Stacked Bars', fontsize=16, y=1.02)
plt.tight_layout()
plt.show()
```

