In [7]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer

from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import silhouette_score, davies_bouldin_score
```

In [8]:
```
# 1. Загрузка данных
train_df = pd.read_csv("Sources/heart_adapt_train.csv")
test_df = pd.read_csv("Sources/heart_adapt_test.csv")
```

In [9]:
```
# Описание структуры набора данных
def describe_data(df):
    print("Описание структуры набора данных:")
    print(df.describe(include='all'))
    for column in df.columns:
        print(f"\nАтрибут: {column}")
        print(f"Текстовое описание: {column}")
        print(f"Цифровое описание:\n{df[column].describe()}")
        plt.figure(figsize=(10, 6))
        sns.histplot(df[column], kde=False)
        plt.title(f'Графическое описание: {column}')
        plt.show()
        plt.close()

describe_data(train_df)
```

Описание структуры набора данных:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS
count	533.000000	589	589	588.000000	462.000000	589.000000
unique	NaN	2	4	NaN	NaN	NaN
top	NaN	M	ASY	NaN	NaN	NaN
freq	NaN	477	348	NaN	NaN	NaN
mean	54.195122	NaN	NaN	133.358844	245.632035	0.258065
std	9.532661	NaN	NaN	18.851852	58.599184	0.437942
min	28.000000	NaN	NaN	80.000000	85.000000	0.000000
25%	48.000000	NaN	NaN	120.000000	209.000000	0.000000
50%	55.000000	NaN	NaN	130.000000	240.000000	0.000000
75%	61.000000	NaN	NaN	144.000000	279.750000	1.000000
max	77.000000	NaN	NaN	200.000000	603.000000	1.000000

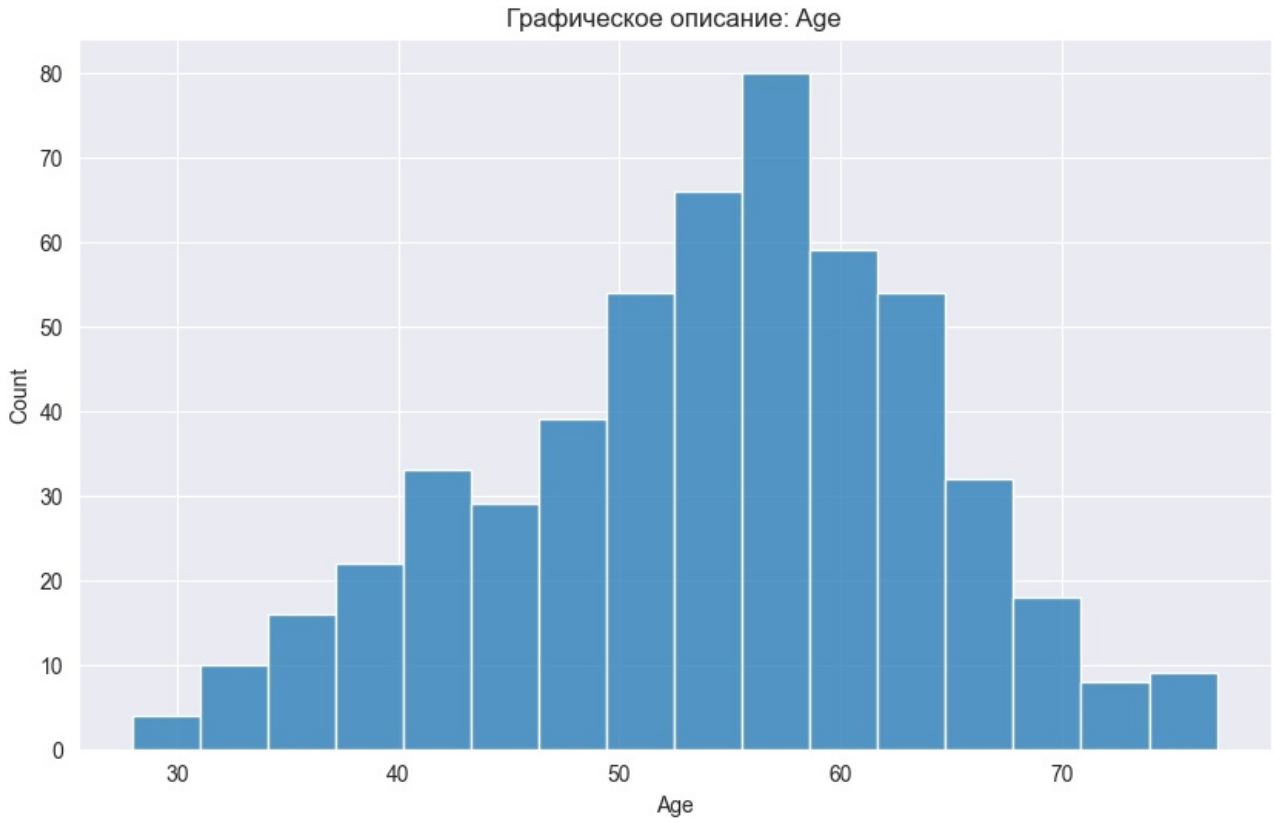
	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
count	589	589.000000	589	589.000000	589
unique	3	NaN	2	NaN	3
top	Normal	NaN	N	NaN	Flat
freq	345	NaN	325	NaN	325
mean	NaN	134.893039	NaN	0.937521	NaN
std	NaN	24.942596	NaN	1.071318	NaN
min	NaN	63.000000	NaN	-2.600000	NaN
25%	NaN	117.000000	NaN	0.000000	NaN
50%	NaN	135.000000	NaN	0.800000	NaN
75%	NaN	154.000000	NaN	1.600000	NaN
max	NaN	195.000000	NaN	5.000000	NaN

	HeartDisease
count	589.000000
unique	NaN
top	NaN
freq	NaN
mean	0.646859
std	0.478352
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

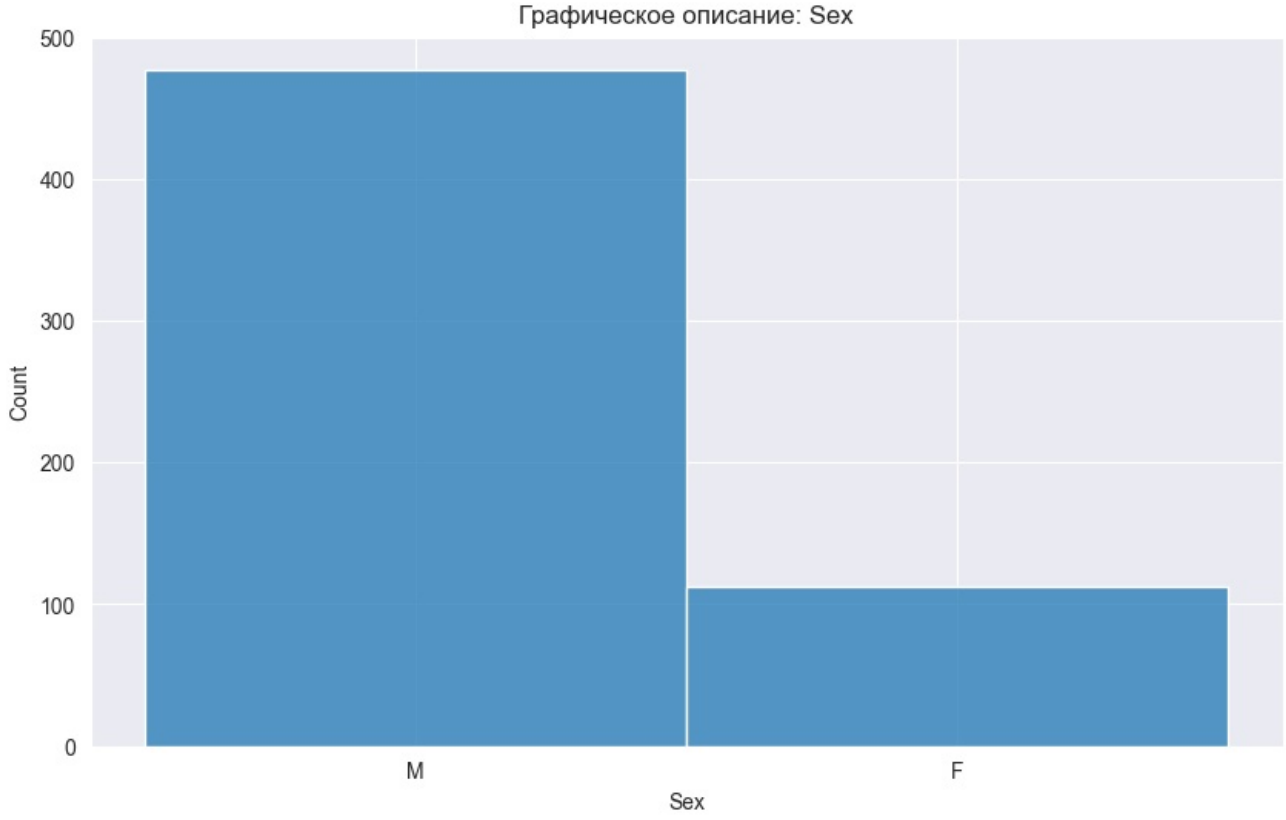
Атрибут: Age
Текстовое описание: Age
Цифровое описание:

count	533.000000
mean	54.195122
std	9.532661
min	28.000000
25%	48.000000
50%	55.000000
75%	61.000000
max	77.000000

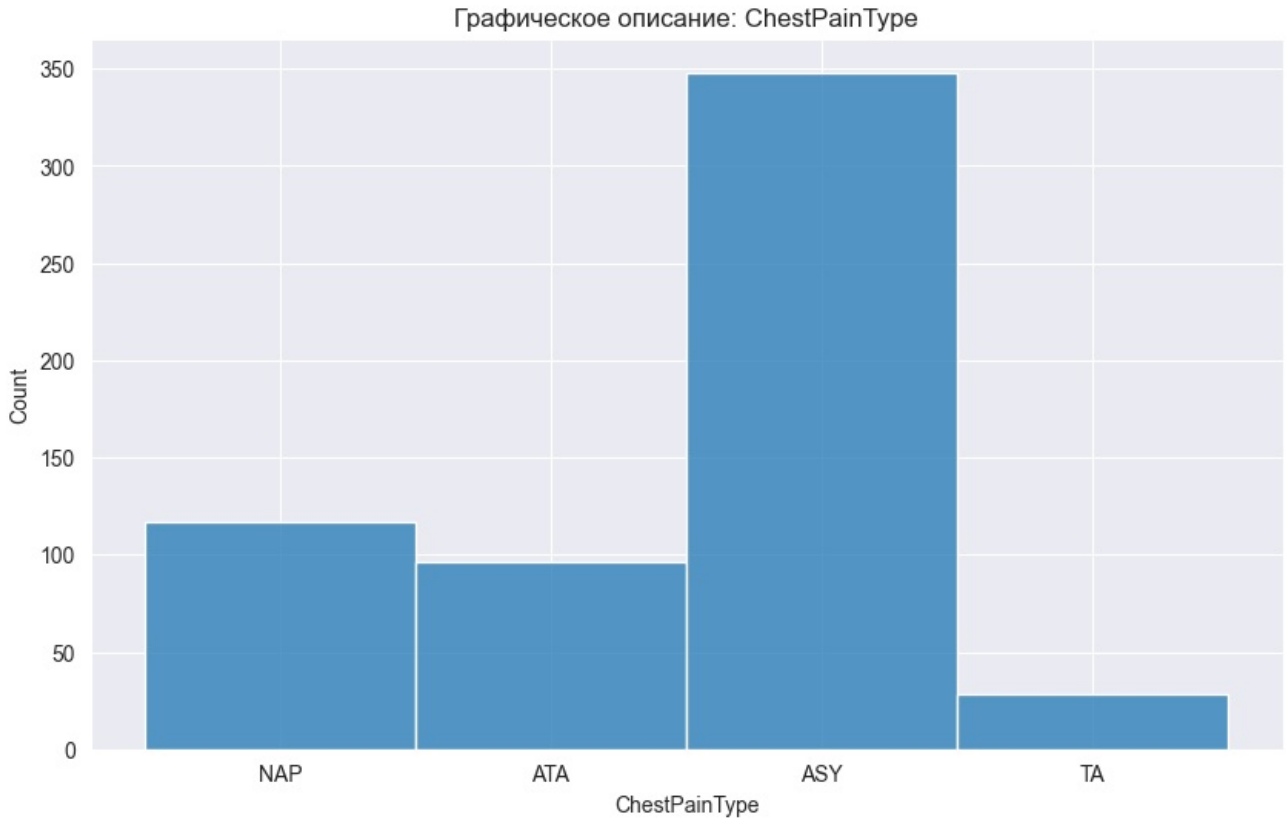
Name: Age, dtype: float64



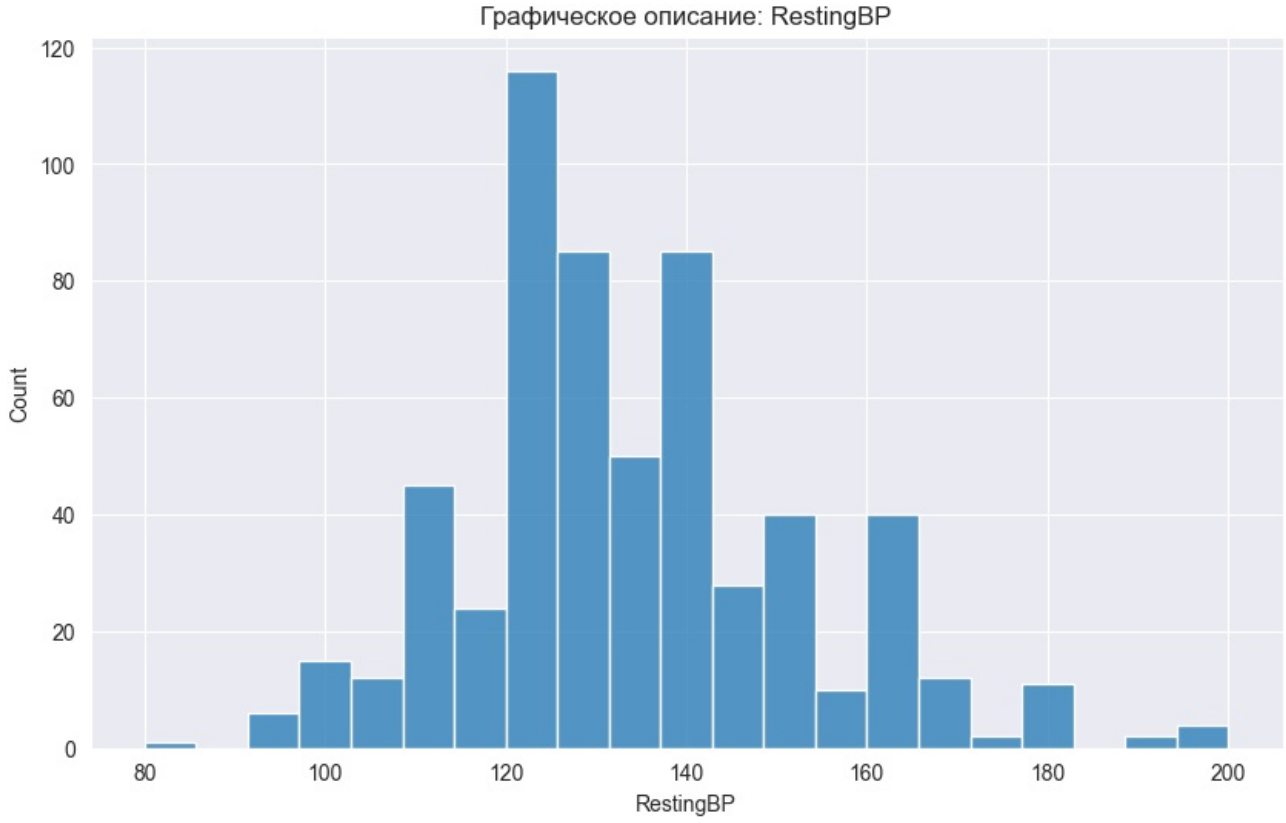

```
Атрибут: Sex
Текстовое описание: Sex
Цифровое описание:
count     589
unique      2
top         M
freq      477
Name: Sex, dtype: object
```



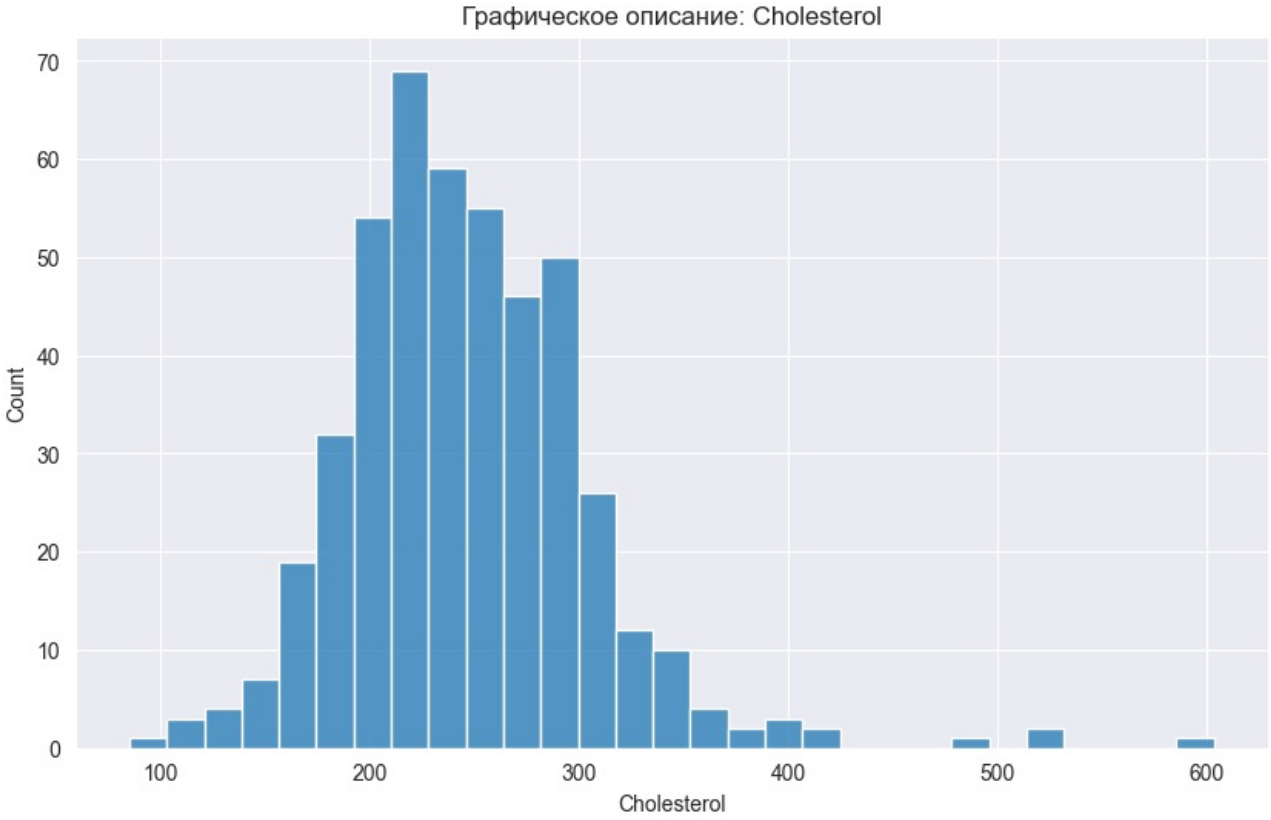

```
Атрибут: ChestPainType
Текстовое описание: ChestPainType
Цифровое описание:
count     589
unique      4
top       ASY
freq      348
Name: ChestPainType, dtype: object
```



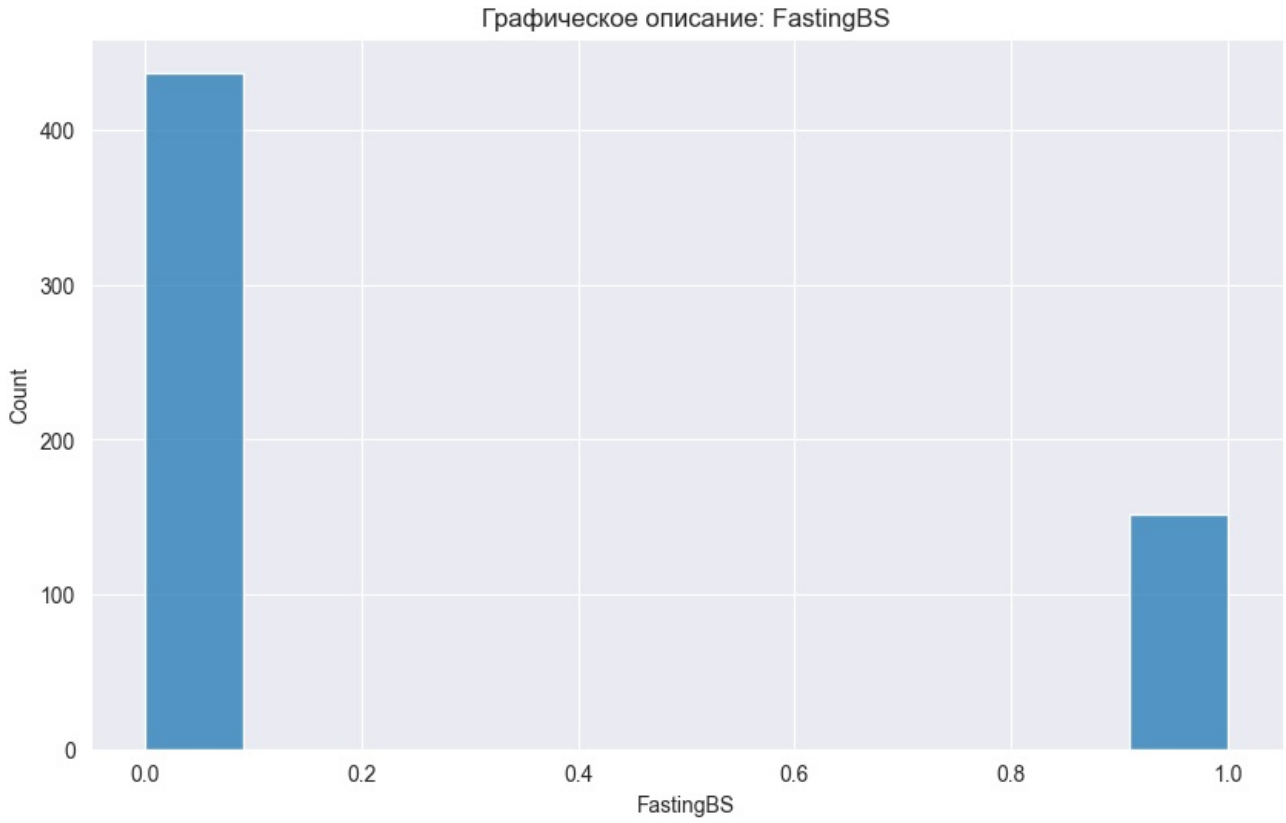

```
Атрибут: RestingBP
Текстовое описание: RestingBP
Цифровое описание:
count    588.000000
mean     133.358844
std       18.851852
min       80.000000
25%      120.000000
50%      130.000000
75%      144.000000
max      200.000000
Name: RestingBP, dtype: float64
```



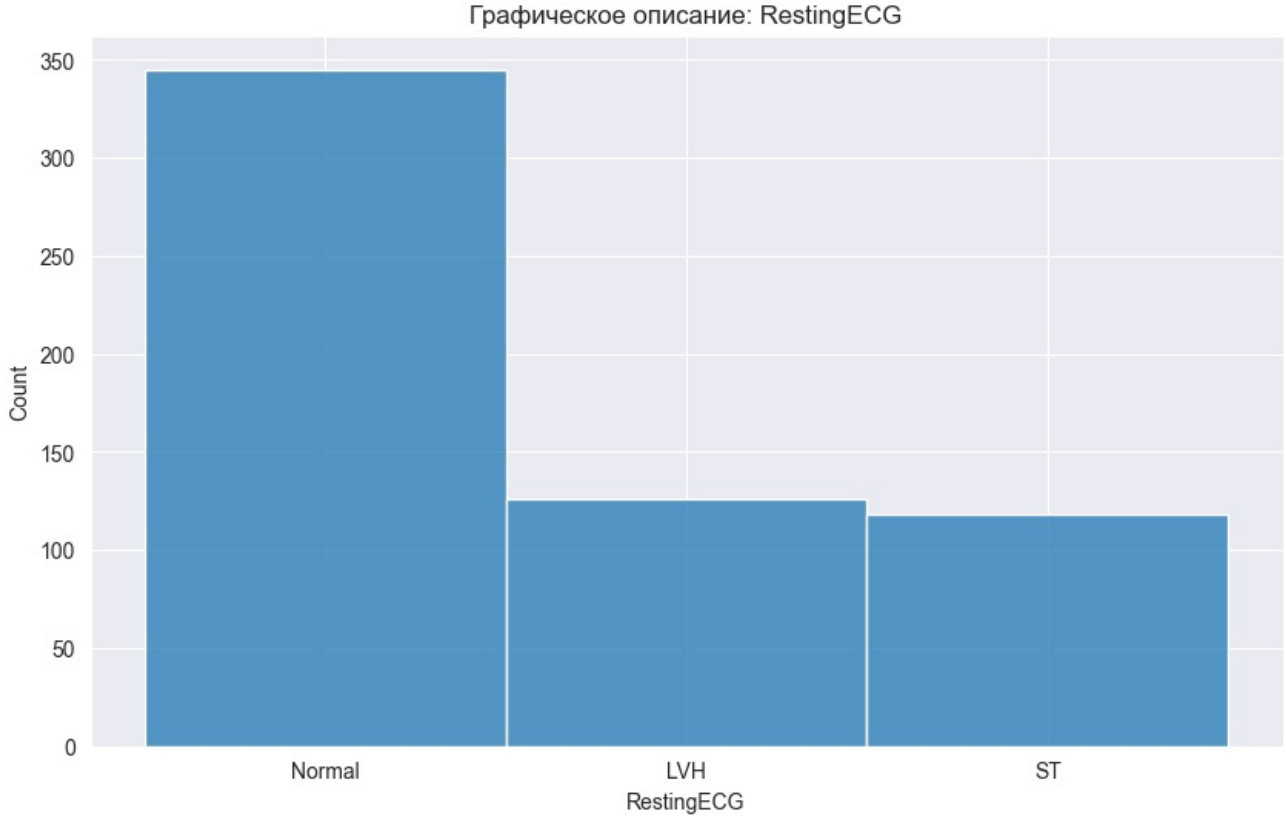
```
Атрибут: Cholesterol
Текстовое описание: Cholesterol
Цифровое описание:
count    462.000000
mean     245.632035
std       58.599184
min       85.000000
25%      209.000000
50%      240.000000
75%      279.750000
max      603.000000
Name: Cholesterol, dtype: float64
```



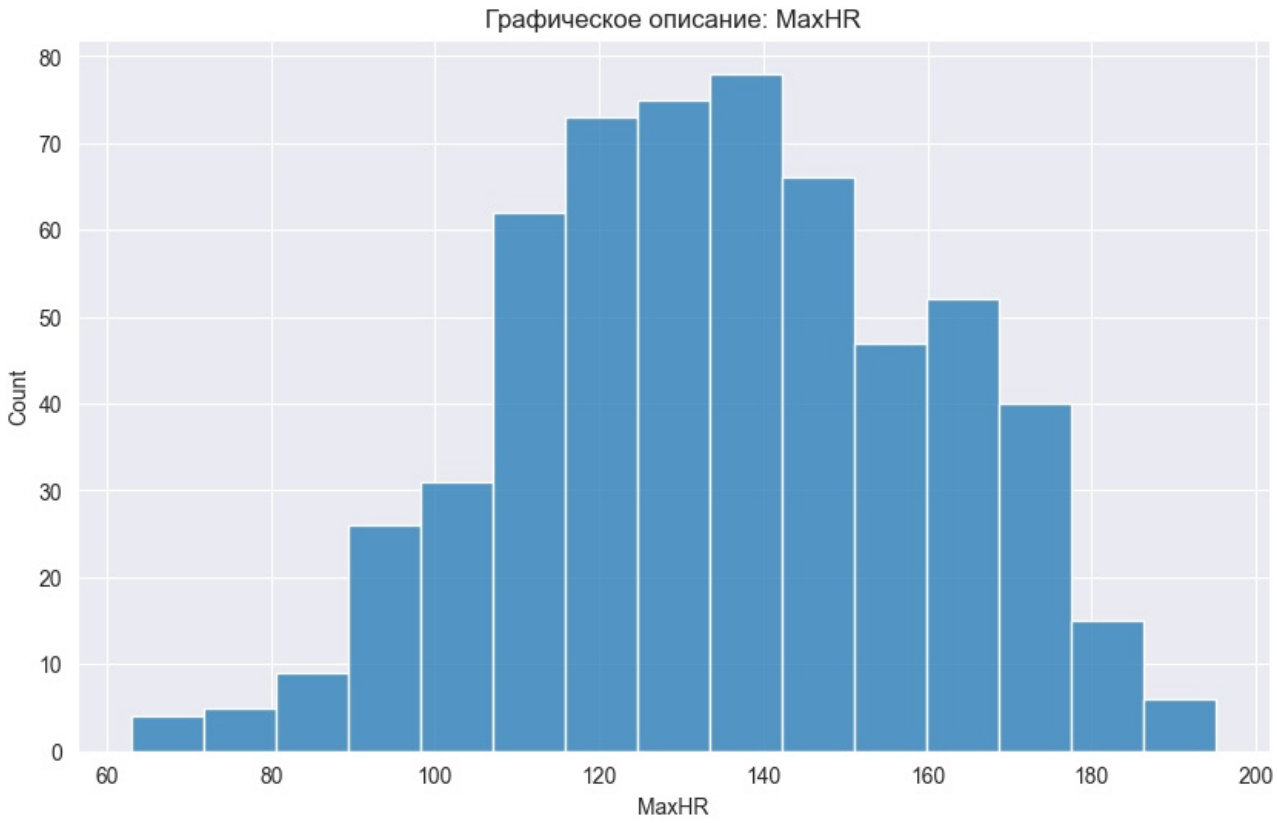

```
Атрибут: FastingBS
Текстовое описание: FastingBS
Цифровое описание:
count    589.000000
mean       0.258065
std        0.437942
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        1.000000
Name: FastingBS, dtype: float64
```



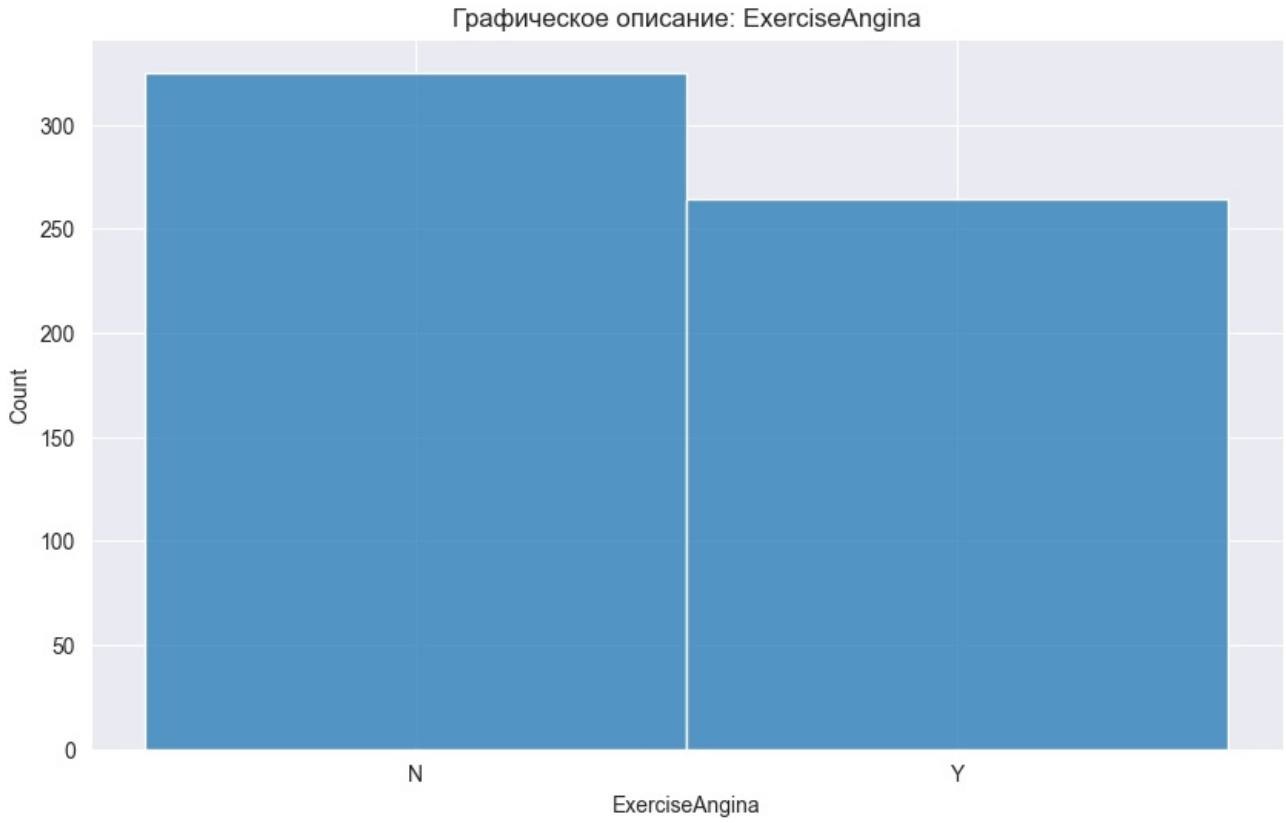

```
Атрибут: RestingECG
Текстовое описание: RestingECG
Цифровое описание:
count        589
unique         3
top       Normal
freq         345
Name: RestingECG, dtype: object
```



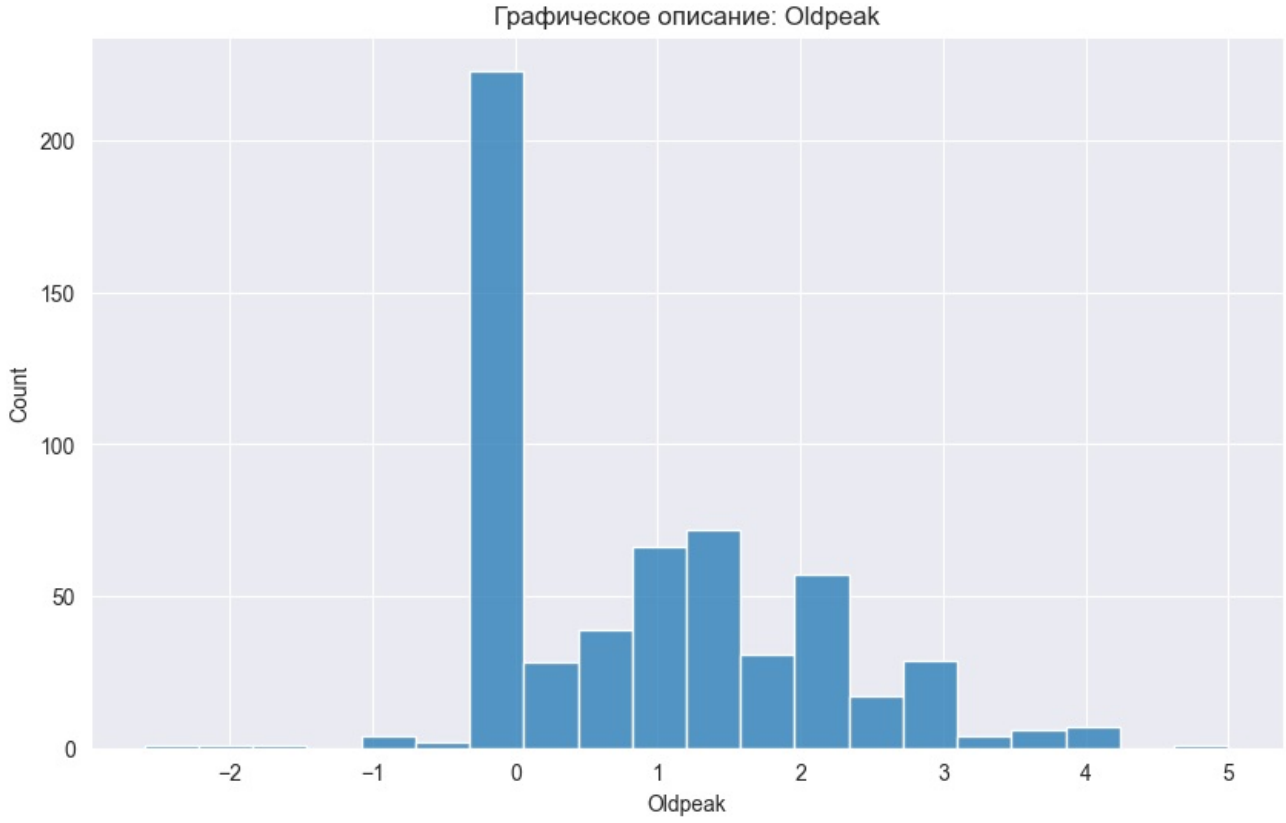
```
Атрибут: MaxHR
Текстовое описание: MaxHR
Цифровое описание:
count    589.000000
mean     134.893039
std       24.942596
min       63.000000
25%      117.000000
50%      135.000000
75%      154.000000
max      195.000000
Name: MaxHR, dtype: float64
```



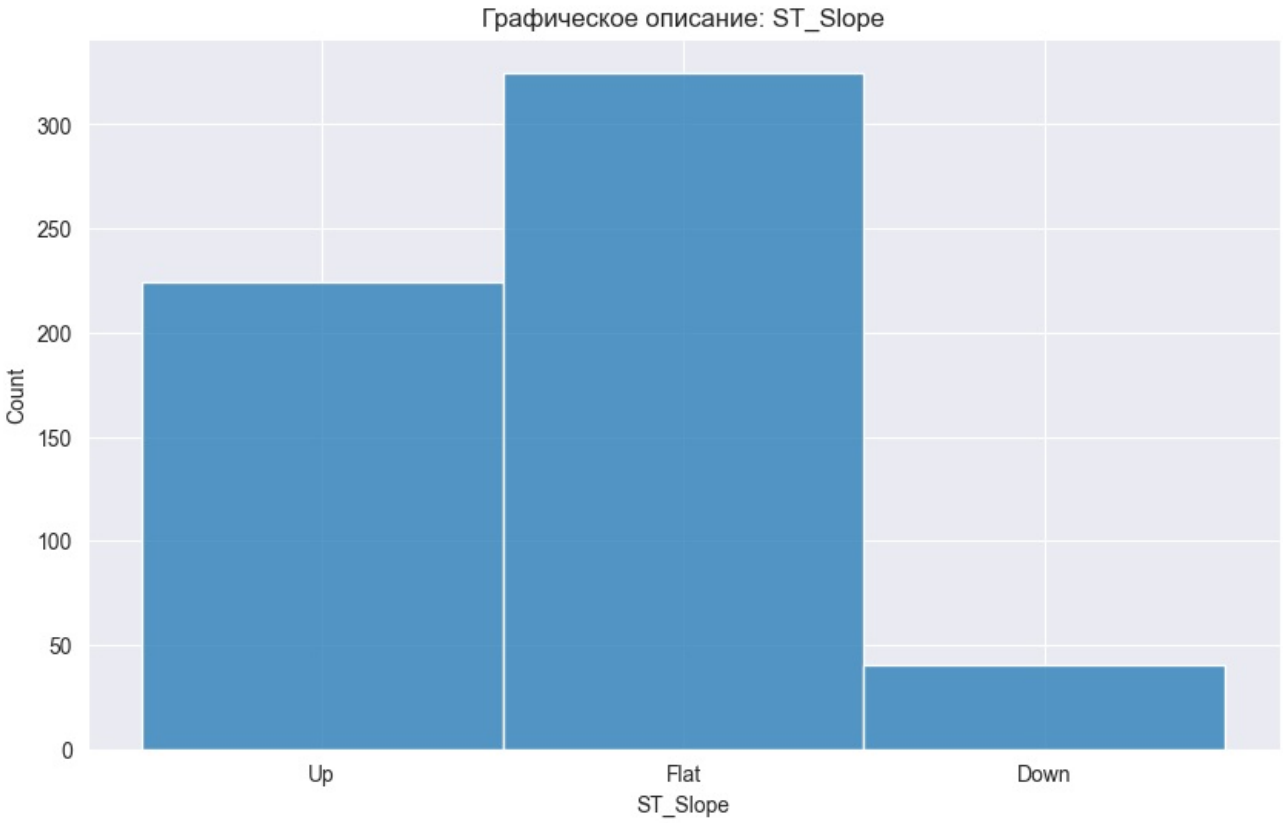

```
Атрибут: ExerciseAngina
Текстовое описание: ExerciseAngina
Цифровое описание:
count     589
unique      2
top         N
freq      325
Name: ExerciseAngina, dtype: object
```



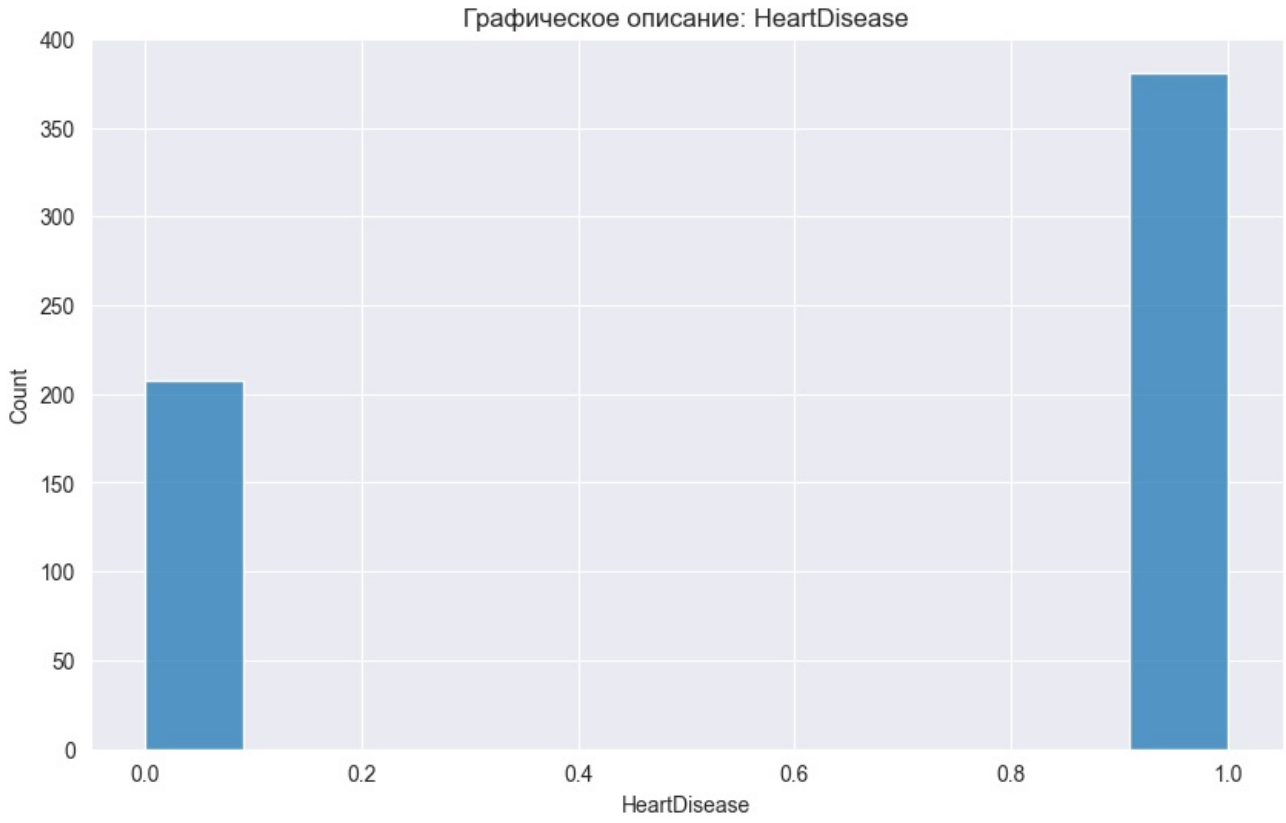

```
Атрибут: Oldpeak
Текстовое описание: Oldpeak
Цифровое описание:
count    589.000000
mean       0.937521
std        1.071318
min       -2.600000
25%        0.000000
50%        0.800000
75%        1.600000
max        5.000000
Name: Oldpeak, dtype: float64
```



```
Атрибут: ST_Slope
Текстовое описание: ST_Slope
Цифровое описание:
count      589
unique       3
top       Flat
freq       325
Name: ST_Slope, dtype: object
```




```
Атрибут: HeartDisease
Текстовое описание: HeartDisease
Цифровое описание:
count    589.000000
mean       0.646859
std        0.478352
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: HeartDisease, dtype: float64
```



In [10]:
```python
# 2. Предобработка данных
def preprocess_data(df, encoder=None, scaler=None):
    numeric_cols = df.select_dtypes(include=[np.number]).columns
    categorical_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']

    numeric_imputer = SimpleImputer(strategy="mean")
    df[numeric_cols] = numeric_imputer.fit_transform(df[numeric_cols])

    categorical_imputer = SimpleImputer(strategy="most_frequent")
    df[categorical_cols] = categorical_imputer.fit_transform(df[categorical_cols])

    if encoder is None:
        encoder = OneHotEncoder(sparse_output=False, drop="first")
        encoded = pd.DataFrame(encoder.fit_transform(df[categorical_cols]))
        encoded.columns = encoder.get_feature_names_out(categorical_cols)
    else:
        encoded = pd.DataFrame(encoder.transform(df[categorical_cols]))
        encoded.columns = encoder.get_feature_names_out(categorical_cols)

    df = df.drop(columns=categorical_cols)
    df = pd.concat([df, encoded], axis=1)

    continuous_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
    if scaler is None:
        scaler = StandardScaler()
        df[continuous_cols] = scaler.fit_transform(df[continuous_cols])
    else:
        df[continuous_cols] = scaler.transform(df[continuous_cols])

    return df, encoder, scaler

train_df, encoder, scaler = preprocess_data(train_df)
test_df, _, _ = preprocess_data(test_df, encoder, scaler)

X_train, y_train = train_df.drop("HeartDisease", axis=1), train_df["HeartDisease"]
X_test, y_test = test_df.drop("HeartDisease", axis=1), test_df["HeartDisease"]
```

In [11]:
```python
# 3.1 Проверка баланса классов
print("Распределение классов в обучающих данных:")
print(y_train.value_counts())

smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
```

```
Распределение классов в обучающих данных:
HeartDisease
1.0    381
0.0    208
Name: count, dtype: int64
```

In [12]:

```
# 3.2 Кластеризация
kmeans = KMeans(n_clusters=3, random_state=42)
dbscan = DBSCAN()
agglomerative = AgglomerativeClustering(n_clusters=3)

labels_kmeans = kmeans.fit_predict(X_train_balanced)
labels_dbscan = dbscan.fit_predict(X_train_balanced)
labels_agglomerative = agglomerative.fit_predict(X_train_balanced)

plt.figure(figsize=(18, 6))

plt.subplot(1, 3, 1)
plt.scatter(X_train_balanced.iloc[:, 0], X_train_balanced.iloc[:, 1], c=labels_kmeans, cmap='viridis')
plt.title('KMeans Clustering')

plt.subplot(1, 3, 2)
plt.scatter(X_train_balanced.iloc[:, 0], X_train_balanced.iloc[:, 1], c=labels_dbscan, cmap='viridis')
plt.title('DBSCAN Clustering')

plt.subplot(1, 3, 3)
plt.scatter(X_train_balanced.iloc[:, 0], X_train_balanced.iloc[:, 1], c=labels_agglomerative, cmap='viridis')
plt.title('Agglomerative Clustering')

plt.show()
plt.close()
```

```
C:\Users\cebot\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```