

```
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")
```

```
df = pd.read_csv('/content/drive/MyDrive/Project 1/diabetes.csv')
```

```
df.head()
```

	id	chol	stab.glu	hdl	ratio	glyhb	location	age	gender	height	weight	frame	bp.1s	bp.1d	bp.2s	bp.2d
0	1000	203.0	82	56.0	3.6	4.31	Buckingham	46	female	62.0	121.0	medium	118.0	59.0	NaN	NaN
1	1001	165.0	97	24.0	6.9	4.44	Buckingham	29	female	64.0	218.0	large	112.0	68.0	NaN	NaN
2	1002	228.0	92	37.0	6.2	4.64	Buckingham	58	female	61.0	256.0	large	190.0	92.0	185.0	92.0
3	1003	78.0	93	12.0	6.5	4.63	Buckingham	67	male	67.0	119.0	large	110.0	50.0	NaN	NaN
4	1005	249.0	90	28.0	8.9	7.72	Buckingham	64	male	68.0	183.0	medium	138.0	80.0	NaN	NaN

Next steps:

[Generate code with df](#)

[New interactive sheet](#)

```
df.dtypes
```

	0
id	int64
chol	float64
stab.glu	int64
hdl	float64
ratio	float64
glyhb	float64
location	object
age	int64
gender	object
height	float64
weight	float64
frame	object
bp.1s	float64
bp.1d	float64
bp.2s	float64
bp.2d	float64
waist	float64
hip	float64
time.ppn	float64

dtype: object

```
df.columns
```

```
Index(['id', 'chol', 'stab.glu', 'hdl', 'ratio', 'glyhb', 'location', 'age',
      'gender', 'height', 'weight', 'frame', 'bp.1s', 'bp.1d', 'bp.2s',
```

```
'bp.2d', 'waist', 'hip', 'time.ppn'],
dtype='object')
```

```
df.describe()
```

	id	chol	stab.glu	hdl	ratio	glyhb	age	height	weight	bp.1s
count	403.000000	402.000000	403.000000	402.000000	402.000000	390.000000	403.000000	398.000000	402.000000	398.000000
mean	15978.310174	207.845771	106.672457	50.445274	4.521642	5.589769	46.851117	66.020101	177.592040	136.904500
std	11881.122124	44.445557	53.076655	17.262626	1.727886	2.242595	16.312333	3.918515	40.340666	22.741000
min	1000.000000	78.000000	48.000000	12.000000	1.500000	2.680000	19.000000	52.000000	99.000000	90.000000
25%	4792.500000	179.000000	81.000000	38.000000	3.200000	4.380000	34.000000	63.000000	151.000000	121.250000
50%	15766.000000	204.000000	89.000000	46.000000	4.200000	4.840000	45.000000	66.000000	172.500000	136.000000
75%	20336.000000	230.000000	106.000000	59.000000	5.400000	5.600000	60.000000	69.000000	200.000000	146.750000
max	41756.000000	443.000000	385.000000	120.000000	19.299999	16.110001	92.000000	76.000000	325.000000	250.000000

```
df.shape
```

```
(403, 19)
```

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

	0
id	0
chol	1
stab.glu	0
hdl	1
ratio	1
glyhb	13
location	0
age	0
gender	0
height	5
weight	1
frame	12
bp.1s	5
bp.1d	5
bp.2s	262
bp.2d	262
waist	2
hip	2
time.ppn	3

```
dtype: int64
```

```
for col in ['chol', 'hdl', 'ratio', 'glyhb', 'height', 'weight', 'bp.1s', 'bp.1d', 'waist', 'hip', 'time.ppn']:
    if col in df.columns:
        df[col] = df[col].fillna(df[col].median())
```

```
if 'frame' in df.columns:
    df['frame'] = df['frame'].fillna(df['frame'].mode()[0])
```

```
print('\nMissing values count after imputation:')
display(df.isnull().sum()[df.isnull().sum() > 0])
```

Missing values count after imputation:

```

0
bp.2s  262
bp.2d  262

dtype: int64
```

```
numerical_cols_for_outliers = [
    'chol', 'stab.glu', 'hdl', 'ratio', 'glyhb', 'height', 'weight',
    'bp.1s', 'bp.1d', 'waist', 'hip', 'time.ppn'
]

def cap_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])
    return df

for col in numerical_cols_for_outliers:
    if col in df.columns:
        df = cap_outliers_iqr(df, col)
        print(f"Outliers capped for column: {col}")

print("\nOutlier handling complete for specified numerical columns.")
```

```
Outliers capped for column: chol
Outliers capped for column: stab.glu
Outliers capped for column: hdl
Outliers capped for column: ratio
Outliers capped for column: glyhb
Outliers capped for column: height
Outliers capped for column: weight
Outliers capped for column: bp.1s
Outliers capped for column: bp.1d
Outliers capped for column: waist
Outliers capped for column: hip
Outliers capped for column: time.ppn

Outlier handling complete for specified numerical columns.
```

```
print('Checking for remaining missing values:')
remaining_missing_values = df.isnull().sum()
display(remaining_missing_values[remaining_missing_values > 0])
```

Checking for remaining missing values:

```

0
bp.2s  262
bp.2d  262

dtype: int64
```

```
as sns
lib.pyplot as plt

"whitegrid")

ing histograms for numerical columns...)
ot = ['chol', 'stab.glu', 'hdl', 'ratio', 'glyhb', 'age', 'height', 'weight', 'bp

size=(20, 15))
enumerate(num_cols_to_plot):
df.columns:
bplot(4, 4, i + 1)
stplot(df[col], kde=True, bins=30)
tle(f'Distribution of {col}')
abel(col)
abel('Frequency')
ut()
```

```
ating count plots for categorical columns...")
ot = ['gender', 'frame', 'location']

size=(15, 5))
enumerate(cat_cols_to_plot):
df.columns:
bplot(1, 3, i + 1)
untplot(data=df, x=col)
tle(f'Count of {col}')
abel(col)
abel('Count')
icks(rotation=45, ha='right')
ut()
```

Generating histograms for numerical columns...



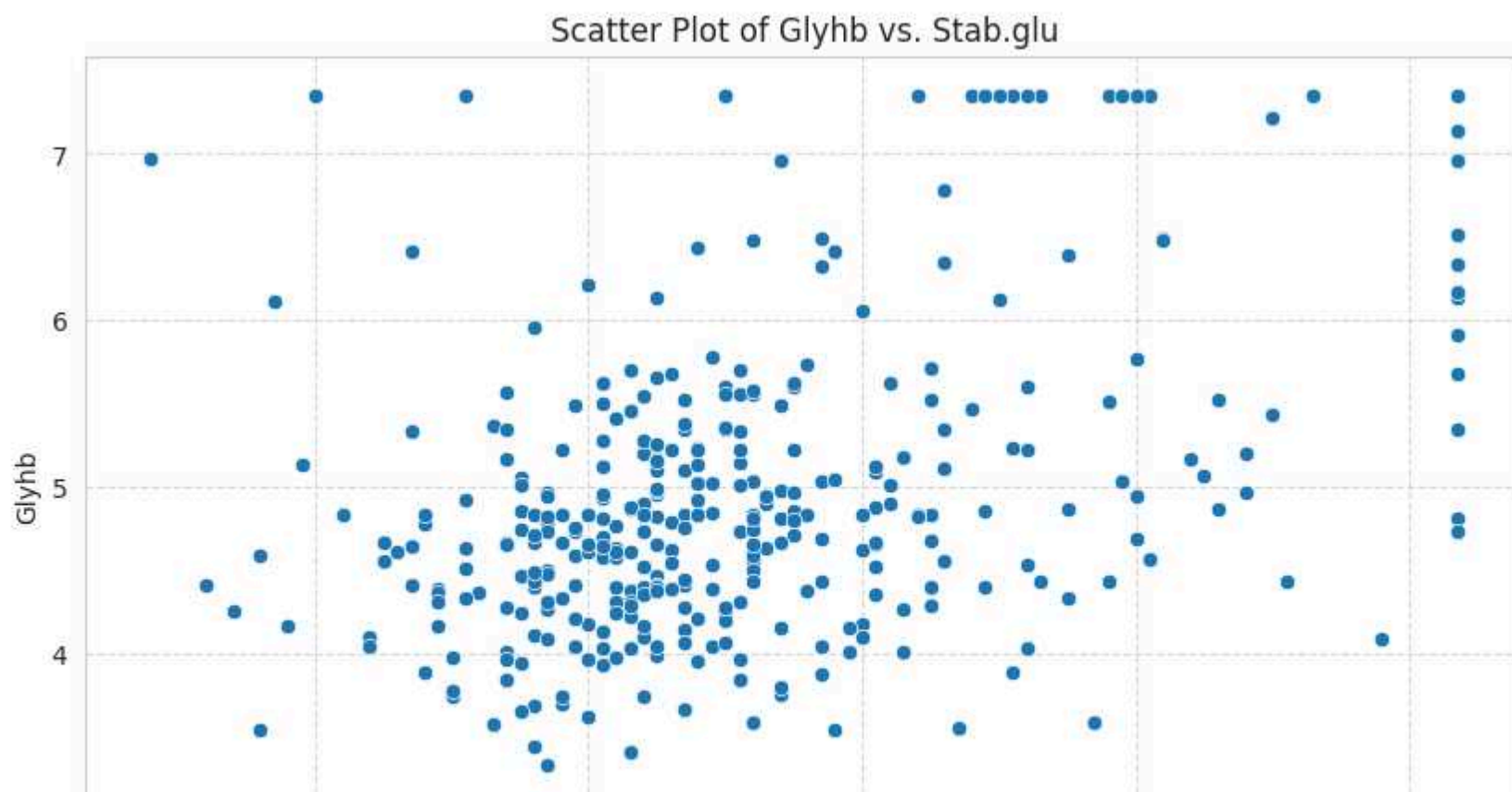
Generating count plots for categorical columns...

Count of gender

Count of frame

Count of location

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='stab.glu', y='glyhb')
plt.title('Scatter Plot of Glyhb vs. Stab.glu')
plt.xlabel('Stab.glu')
plt.ylabel('Glyhb')
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```



```
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.boxplot(data=df, x='gender', y='glyhb', palette='viridis')
plt.title('Glyhb Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Glyhb')
plt.grid(True, linestyle='--', alpha=0.7)

plt.subplot(1, 2, 2)
sns.boxplot(data=df, x='gender', y='stab.glu', palette='viridis')
plt.title('Stab.glu Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Stab.glu')
plt.grid(True, linestyle='--', alpha=0.7)

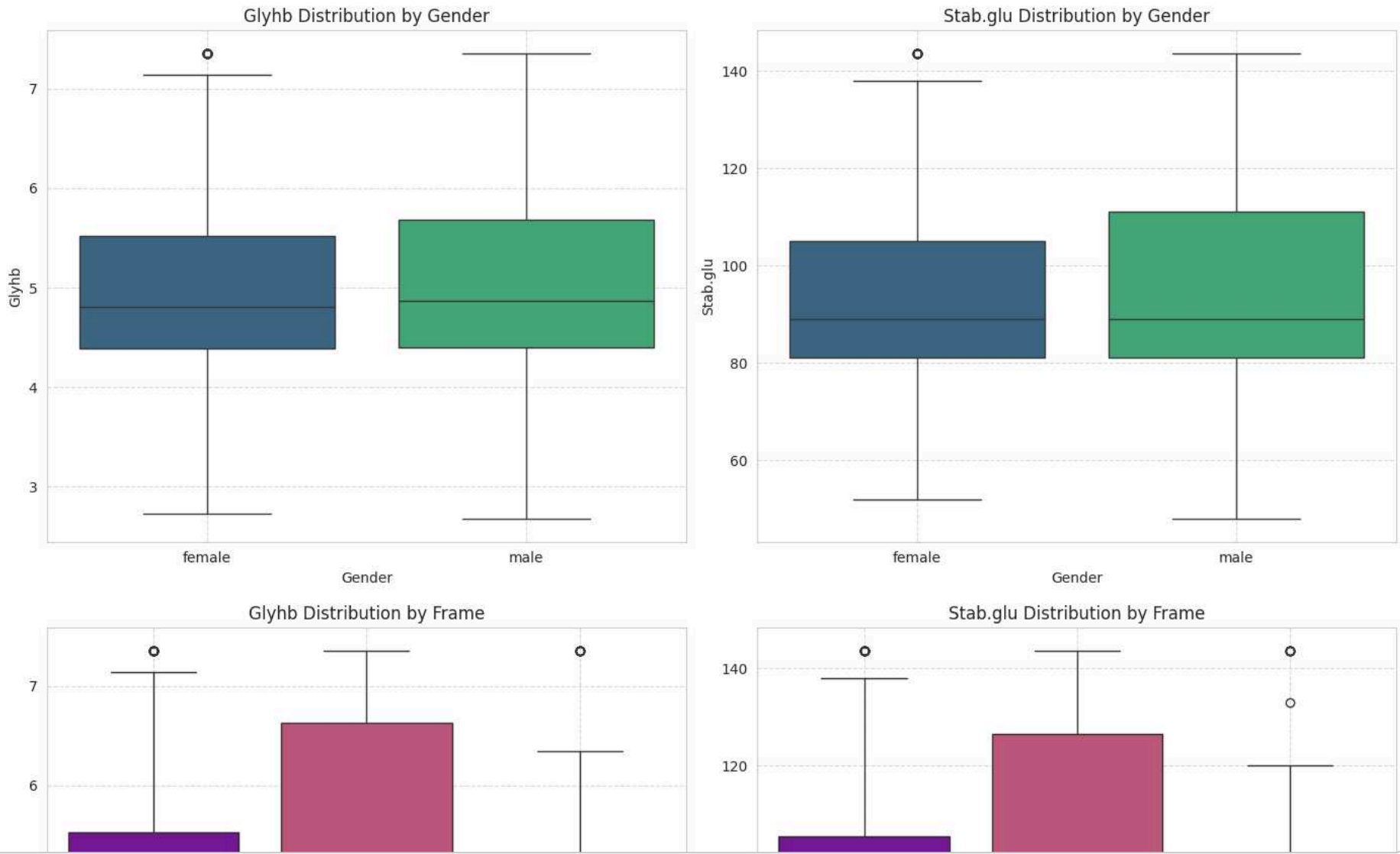
plt.tight_layout()
plt.show()

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.boxplot(data=df, x='frame', y='glyhb', palette='plasma')
plt.title('Glyhb Distribution by Frame')
plt.xlabel('Frame')
plt.ylabel('Glyhb')
plt.grid(True, linestyle='--', alpha=0.7)

plt.subplot(1, 2, 2)
sns.boxplot(data=df, x='frame', y='stab.glu', palette='plasma')
plt.title('Stab.glu Distribution by Frame')
plt.xlabel('Frame')
plt.ylabel('Stab.glu')
plt.grid(True, linestyle='--', alpha=0.7)

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



```
selected_cols = ['glyhb', 'stab.glu', 'age', 'chol']

plt.figure(figsize=(10, 8))
sns.pairplot(df[selected_cols])
plt.suptitle('Pairplot of Glyhb, Stab.glu, Age, and Cholesterol', y=1.02) # Adjust supitle position
plt.tight_layout()
plt.show()
```

<Figure size 1000x800 with 0 Axes>



```

print("\nGenerating correlation heatmap for all numerical features...")

plt.figure(figsize=(15, 12))

numerical_cols = df.select_dtypes(include=np.number).columns.tolist()

if 'id' in numerical_cols:
    numerical_cols.remove('id')

correlation_matrix = df[numerical_cols].corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title('Correlation Matrix of All Numerical Features')

plt.savefig('correlation_heatmap_all_features.png', dpi=300, bbox_inches='tight')

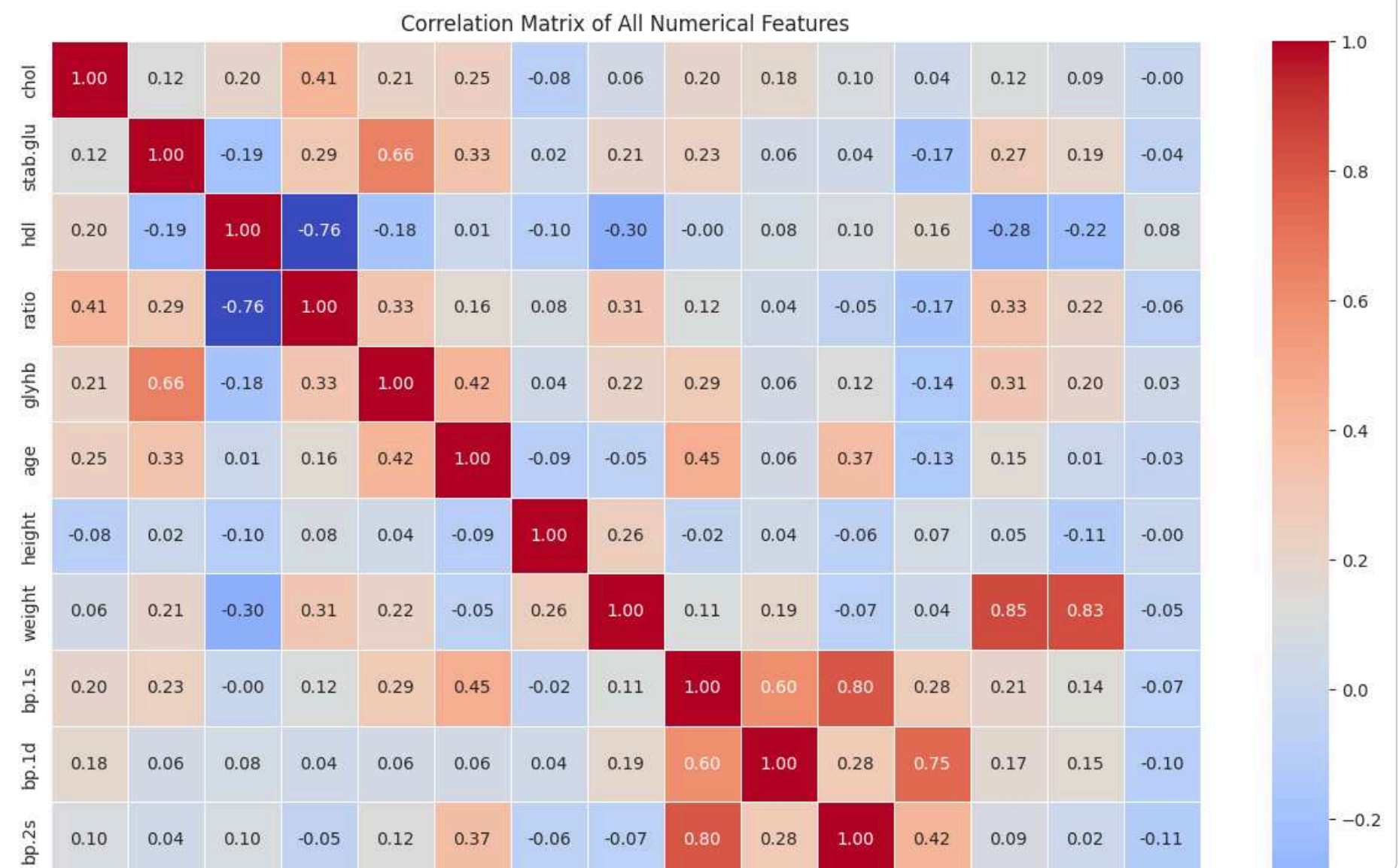
plt.show()

plt.close()

glyhb_correlations = correlation_matrix['glyhb'].sort_values(ascending=False)

```

Generating correlation heatmap for all numerical features...



```

top_5_correlated_features = glyhb_correlations.drop('glyhb').head(5)

print("\nTop 5 features most correlated with 'glyhb':")
display(top_5_correlated_features)

features_for_heatmap = top_5_correlated_features.index.tolist() + ['glyhb']

print("\nGenerating subset correlation heatmap...")

plt.figure(figsize=(10, 8))
correlation_subset_matrix = df[features_for_heatmap].corr()
sns.heatmap(correlation_subset_matrix, annot=True, cmap='coolwarm', fmt=".2f", li
plt.title('Correlation Matrix of Glyhb and its Top 5 Correlated Features')
plt.savefig('correlation_heatmap_top5_glyhb.png', dpi=300, bbox_inches='tight')
plt.show()
plt.close()

```

Top 5 features most correlated with 'glyhb':

	glyhb
stab.glu	0.656174
age	0.418364
ratio	0.326092
waist	0.310004
bp.1s	0.288802

dtype: float64

Generating subset correlation heatmap...

