In [1]:
```
import pandas as pd

#data = pd.read_csv('../cirrhosis.csv')
#data
#data.sample(200).to_csv('sample.csv', index=False)
```

In [2]:
```
sampled_data = pd.read_csv('sample.csv')
sampled_data.isnull().sum()
sampled_data
```

Out[2]:

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	S
0	411	1119	C	NaN	18628	F	NaN	NaN	
1	100	552	D	Placebo	18799	M	N	Y	
2	367	2202	C	NaN	23376	F	NaN	NaN	
3	58	4459	C	D-penicillamine	16279	M	N	N	
4	350	662	D	NaN	17532	F	NaN	NaN	
...	...	...	...	...	...	...	...	...	
195	119	515	D	D-penicillamine	19817	F	N	N	
196	136	3098	C	D-penicillamine	20440	F	N	N	
197	39	2297	D	D-penicillamine	20232	F	N	Y	
198	269	1363	C	Placebo	16467	F	N	N	
199	221	2050	C	Placebo	20684	F	N	Y	

200 rows × 20 columns

In [3]:
```
sampled_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   ID             200 non-null    int64  
 1   N_Days         200 non-null    int64  
 2   Status         200 non-null    object 
 3   Drug           152 non-null    object 
 4   Age            200 non-null    int64  
 5   Sex            200 non-null    object 
 6   Ascites        152 non-null    object 
 7   Hepatomegaly   152 non-null    object 
 8   Spiders        152 non-null    object 
 9   Edema          200 non-null    object 
 10  Bilirubin      200 non-null    float64
 11  Cholesterol    139 non-null    float64
 12  Albumin        200 non-null    float64
 13  Copper         152 non-null    float64
 14  Alk_Phos       152 non-null    float64
 15  SGOT           152 non-null    float64
 16  Tryglicerides  137 non-null    float64
 17  Platelets      196 non-null    float64
 18  Prothrombin    198 non-null    float64
 19  Stage          199 non-null    float64
dtypes: float64(10), int64(3), object(7)
memory usage: 31.4+ KB
```

In [4]:
```
sampled_data.describe()
```

Out[4]:

	ID	N_Days	Age	Bilirubin	Cholesterol	Albu
count	200.000000	200.000000	200.000000	200.000000	139.000000	200.00
mean	211.320000	1914.175000	18678.670000	3.347500	389.071942	3.47
std	118.428944	1092.006274	3790.299455	4.426246	268.640044	0.43
min	1.000000	41.000000	9598.000000	0.300000	168.000000	1.96
25%	112.750000	1113.000000	16065.750000	0.800000	258.500000	3.19
50%	213.000000	1690.000000	18628.000000	1.450000	318.000000	3.50
75%	306.000000	2573.250000	21431.750000	3.500000	395.000000	3.76
max	418.000000	4509.000000	28650.000000	28.000000	1775.000000	4.52

In [5]:
```
sampled_data["Age"].fillna(sampled_data["Age"].mean(), inplace=True)
sampled_data["Bilirubin"].fillna(sampled_data["Bilirubin"].mean(), inplace=T
sampled_data["Copper"].fillna(sampled_data["Copper"].mean(), inplace=True)
sampled_data["Alk_Phos"].fillna(sampled_data["Alk_Phos"].mean(), inplace=Tru
sampled_data["Tryglicerides"].fillna(sampled_data["Tryglicerides"].mean(), i
sampled_data["Platelets"].fillna(sampled_data["Platelets"].mean(), inplace=T
sampled_data["Prothrombin"].fillna(sampled_data["Prothrombin"].mean(), inpla
sampled_data["Stage"].fillna(sampled_data["Stage"].mean(), inplace=True)
sampled_data["SGOT"].fillna(sampled_data["SGOT"].mean(), inplace=True)
sampled_data["Cholesterol"].fillna(sampled_data["Cholesterol"].mean(), inpla
```

```
sampled_data
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:1: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Age"].fillna(sampled_data["Age"].mean(), inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:2: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Bilirubin"].fillna(sampled_data["Bilirubin"].mean(), inplace
=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:3: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Copper"].fillna(sampled_data["Copper"].mean(), inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:4: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Alk_Phos"].fillna(sampled_data["Alk_Phos"].mean(), inplace=T
rue)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:5: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
```

```
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Triglicerides"].fillna(sampled_data["Triglicerides"].mean(),
inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:6: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Platelets"].fillna(sampled_data["Platelets"].mean(), inplace
=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:7: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Prothrombin"].fillna(sampled_data["Prothrombin"].mean(), inp
lace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:8: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Stage"].fillna(sampled_data["Stage"].mean(), inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:9: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
```

```
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["SGOT"].fillna(sampled_data["SGOT"].mean(), inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\3091490043.py:10: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through ch
ained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  sampled_data["Cholesterol"].fillna(sampled_data["Cholesterol"].mean(), inp
lace=True)
```

Out[5]:

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	S
0	411	1119	C	NaN	18628	F	NaN	NaN	
1	100	552	D	Placebo	18799	M	N	Y	
2	367	2202	C	NaN	23376	F	NaN	NaN	
3	58	4459	C	D-penicillamine	16279	M	N	N	
4	350	662	D	NaN	17532	F	NaN	NaN	
...	...	...	...	...	...	...	...	...	
195	119	515	D	D-penicillamine	19817	F	N	N	
196	136	3098	C	D-penicillamine	20440	F	N	N	
197	39	2297	D	D-penicillamine	20232	F	N	Y	
198	269	1363	C	Placebo	16467	F	N	N	
199	221	2050	C	Placebo	20684	F	N	Y	

200 rows × 20 columns

In [6]:
```
sampled_data.isnull().sum()
```

Out[6]:
```
ID                0
N_Days            0
Status            0
Drug             48
Age               0
Sex               0
Ascites          48
Hepatomegaly     48
Spiders          48
Edema             0
Bilirubin         0
Cholesterol       0
Albumin           0
Copper            0
Alk_Phos          0
SGOT              0
Trygliceride      0
Platelets         0
Prothrombin       0
Stage             0
dtype: int64
```

In [7]:
```
sampled_data.fillna(method='bfill', inplace=True)
sampled_data
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_7324\2206756072.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future v
ersion. Use obj.ffill() or obj.bfill() instead.
  sampled_data.fillna(method='bfill', inplace=True)
```

Out[7]:

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	S
0	411	1119	C	Placebo	18628	F	N	Y	
1	100	552	D	Placebo	18799	M	N	Y	
2	367	2202	C	D-penicillamine	23376	F	N	N	
3	58	4459	C	D-penicillamine	16279	M	N	N	
4	350	662	D	Placebo	17532	F	N	Y	
...	...	...	...	...	...	...	...	...	
195	119	515	D	D-penicillamine	19817	F	N	N	
196	136	3098	C	D-penicillamine	20440	F	N	N	
197	39	2297	D	D-penicillamine	20232	F	N	Y	
198	269	1363	C	Placebo	16467	F	N	N	
199	221	2050	C	Placebo	20684	F	N	Y	

200 rows × 20 columns

In [8]:
```
mode_of_sex = sampled_data["Sex"].mode()
mode_of_sex
```

Out[8]:
```
0    F
Name: Sex, dtype: object
```

In [9]:
```
median_of_age = sampled_data["Age"].median()
median_of_age
```

Out[9]:
```
np.float64(18628.0)
```

In [10]:
```
Q1 = sampled_data["N_Days"].quantile(0.25)
Q3 = sampled_data["N_Days"].quantile(0.75)
IQR = Q3 - Q1
Q3, Q1, IQR
```

Out[10]:
```
(np.float64(2573.25), np.float64(1113.0), np.float64(1460.25))
```

In [11]:
```
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR
lower_limit, upper_limit
```

Out[11]:
```
(np.float64(-1077.375), np.float64(4763.625))
```

In [12]:
```
data_no_outliers = sampled_data[sampled_data["N_Days"].between(lower_limit,
data_no_outliers
```

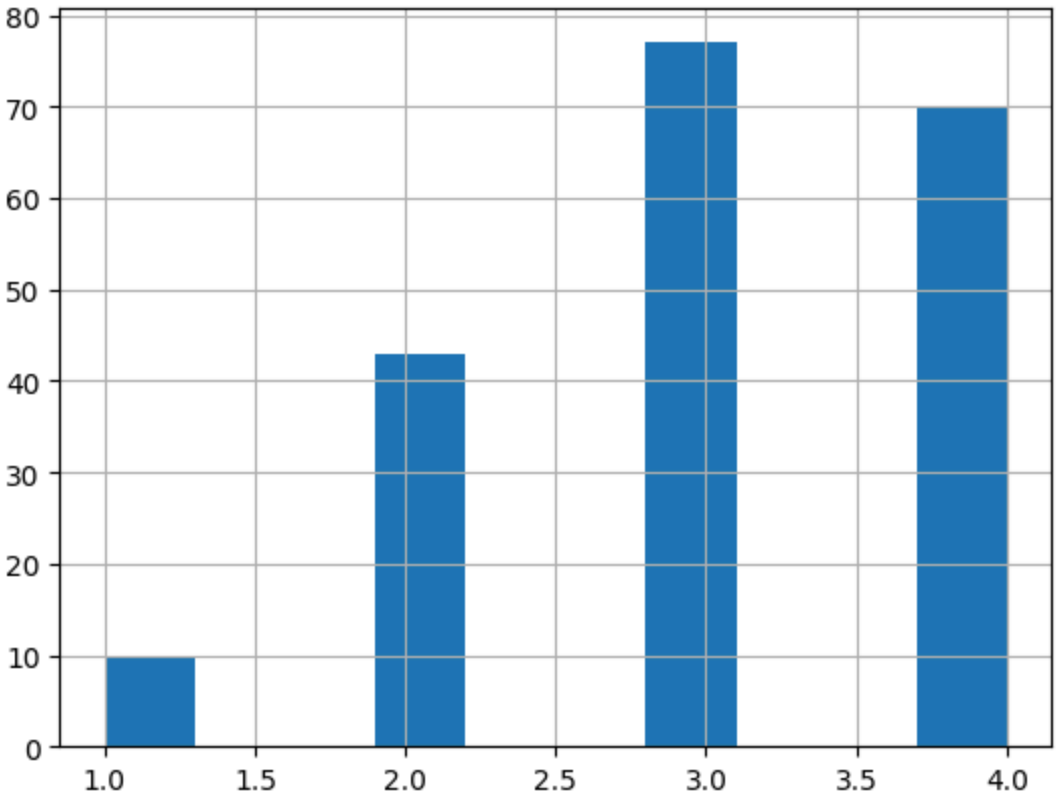
Out[12]:

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	S
0	411	1119	C	Placebo	18628	F	N	Y	
1	100	552	D	Placebo	18799	M	N	Y	
2	367	2202	C	D-penicillamine	23376	F	N	N	
3	58	4459	C	D-penicillamine	16279	M	N	N	
4	350	662	D	Placebo	17532	F	N	Y	
...	...	...	...	...	...	...	...	...	
195	119	515	D	D-penicillamine	19817	F	N	N	
196	136	3098	C	D-penicillamine	20440	F	N	N	
197	39	2297	D	D-penicillamine	20232	F	N	Y	
198	269	1363	C	Placebo	16467	F	N	N	
199	221	2050	C	Placebo	20684	F	N	Y	

200 rows × 20 columns

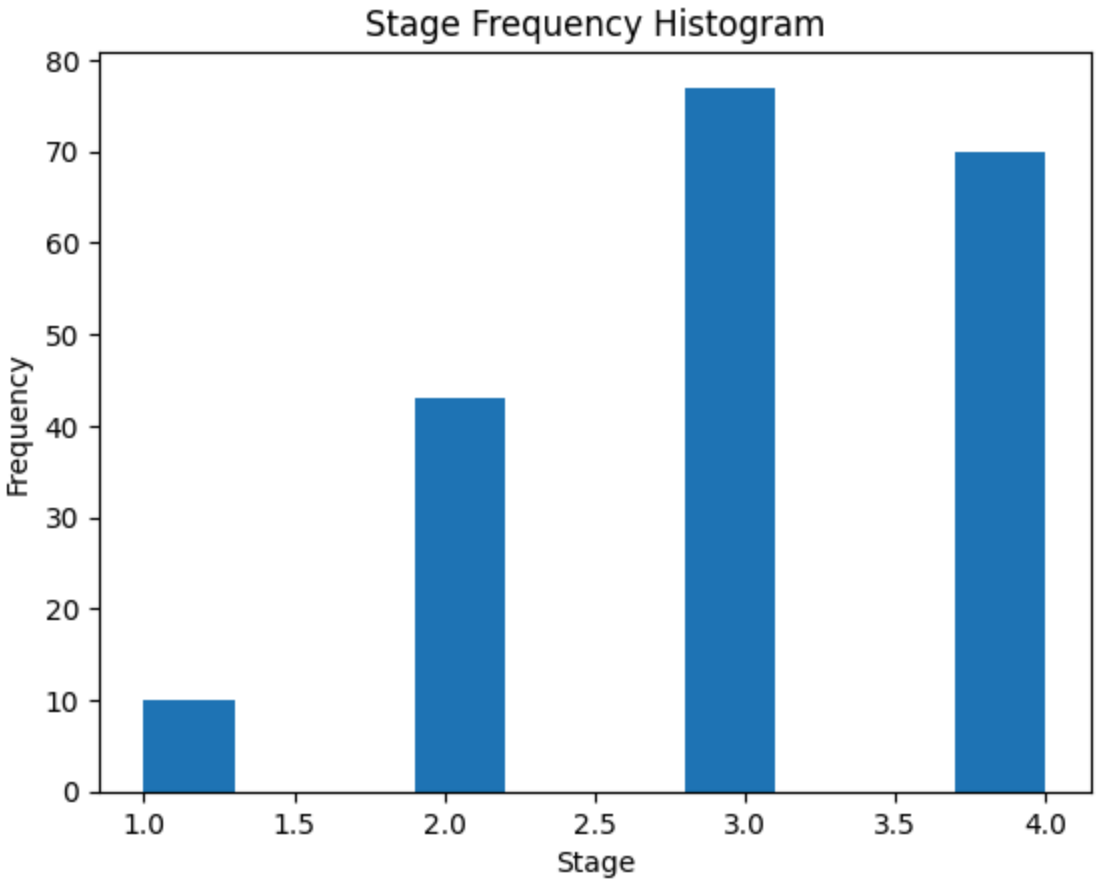
In [13]:
```
sampled_data["Stage"].hist(bins=10)
```

Out[13]: <Axes: >

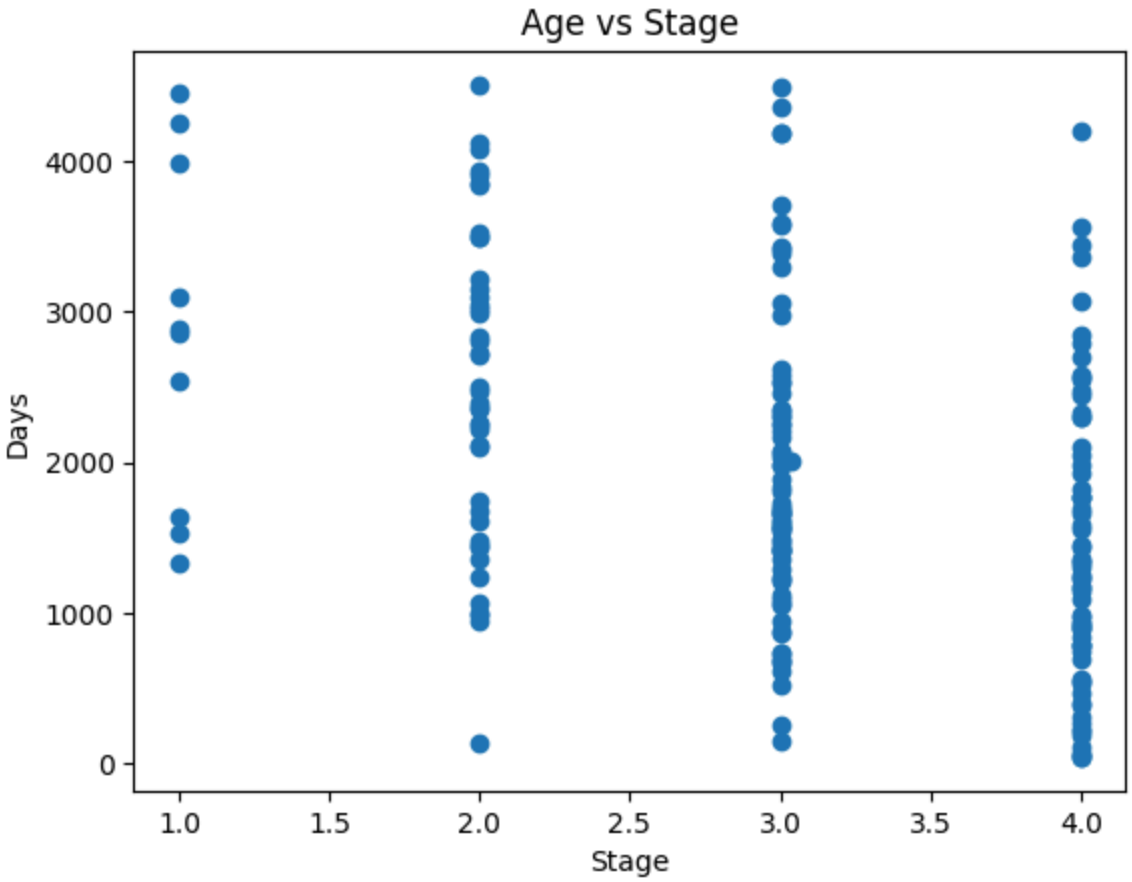


In [14]:
```
import matplotlib.pyplot as plt

plt.hist(sampled_data["Stage"], bins=10)
plt.xlabel("Stage")
plt.ylabel("Frequency")
plt.title("Stage Frequency Histogram")
plt.show()
```



In [15]:
```
plt.scatter(sampled_data["Stage"], sampled_data["N_Days"])
plt.ylabel("Days")
plt.xlabel("Stage")
plt.title("Age vs Stage")
plt.show()
```



In [16]:
```
cleaned_data = sampled_data.drop(columns=["Drug", "Spiders", "Hepatomegaly",
cleaned_data
```

Out[16]:

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper	Alk_P
0	411	1119	18628	0.6	389.071942	3.57	98.907895	1892.353
1	100	552	18799	2.3	178.000000	3.00	145.000000	746.000
2	367	2202	23376	1.1	389.071942	3.49	98.907895	1892.353
3	58	4459	16279	0.7	242.000000	4.08	73.000000	5890.000
4	350	662	17532	2.1	389.071942	4.10	98.907895	1892.353
...	...	...	...	...	...	...	...	
195	119	515	19817	0.6	636.000000	3.83	129.000000	944.000
196	136	3098	20440	0.8	263.000000	3.35	27.000000	1636.000
197	39	2297	20232	0.7	282.000000	3.00	52.000000	9066.800
198	269	1363	16467	3.6	374.000000	3.50	143.000000	1428.000
199	221	2050	20684	0.9	360.000000	3.65	72.000000	3186.000

200 rows × 13 columns

In [17]:
```
cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   ID             200 non-null    int64
 1   N_Days         200 non-null    int64
 2   Age            200 non-null    int64
 3   Bilirubin      200 non-null    float64
 4   Cholesterol    200 non-null    float64
 5   Albumin        200 non-null    float64
 6   Copper         200 non-null    float64
 7   Alk_Phos       200 non-null    float64
 8   SGOT           200 non-null    float64
 9   Tryglicerides  200 non-null    float64
 10  Platelets      200 non-null    float64
 11  Prothrombin    200 non-null    float64
 12  Stage          200 non-null    float64
dtypes: float64(10), int64(3)
memory usage: 20.4 KB
```

In [18]:
```
correlated_data = cleaned_data.corr()
correlated_data
```

Out[18]:

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin
ID	1.000000	-0.316727	0.066185	-0.100034	-0.005703	-0.113123
N_Days	-0.316727	1.000000	-0.144761	-0.403851	-0.108755	0.509910
Age	0.066185	-0.144761	1.000000	-0.027868	-0.151459	-0.195860
Bilirubin	-0.100034	-0.403851	-0.027868	1.000000	0.304989	-0.351782
Cholesterol	-0.005703	-0.108755	-0.151459	0.304989	1.000000	-0.113405
Albumin	-0.113123	0.509910	-0.195860	-0.351782	-0.113405	1.000000
Copper	0.018039	-0.284678	0.023227	0.358370	0.103652	-0.239669
Alk_Phos	-0.209919	0.158817	-0.018283	0.123078	0.187121	-0.071006
SGOT	0.000901	-0.150503	-0.079320	0.372018	0.318303	-0.252584
Tryglicerides	0.010888	-0.175815	-0.053196	0.294586	0.174378	-0.047794
Platelets	-0.047873	0.148136	-0.191305	0.042362	0.139782	0.150114
Prothrombin	-0.310112	-0.180020	0.164127	0.246415	-0.112854	-0.261991
Stage	-0.031836	-0.414424	0.196981	0.182695	-0.016330	-0.268827

In [19]:
```
import seaborn as sns
plt.figure(figsize=(9, 7))
sns.heatmap(correlated_data, annot=True, linewidths=1)
plt.title('Correlation Heatmap')
plt.show()
```