In [1]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
import pandas as pd
import matplotlib. pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
lung_data = pd.read_csv("survey lung cancer.csv")
```

In [4]:
```
lung_data
```

Out[4]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE
0	M	69	1	2	2	1	1
1	M	74	2	1	1	1	2
2	F	59	1	1	1	2	1
3	M	63	2	2	2	1	1
4	F	63	1	2	1	1	1
...	...	...	...	...	...	...	...
304	F	56	1	1	1	2	2
305	M	70	2	1	1	1	1
306	M	58	2	1	1	1	1
307	M	67	2	1	2	1	1
308	M	62	1	1	1	2	1

309 rows × 16 columns

In [5]:
```
lung_data.head()
```

Out[5]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FA
0	M	69	1	2	2	1	1	
1	M	74	2	1	1	1	2	
2	F	59	1	1	1	2	1	
3	M	63	2	2	2	1	1	
4	F	63	1	2	1	1	1	

In [6]:
```
lung_data.tail()
```

Out[6]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE
304	F	56	1	1	1	2	2
305	M	70	2	1	1	1	1
306	M	58	2	1	1	1	1
307	M	67	2	1	2	1	1
308	M	62	1	1	1	2	1

In [7]:
```
#dependent_variable
x = lung_data.iloc[:,0:-1]
print(x)
```

```
    GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  PEER_PRESSURE  \
0        M   69        1               2        2              1
1        M   74        2               1        1              1
2        F   59        1               1        1              2
3        M   63        2               2        2              1
4        F   63        1               2        1              1
..     ...  ...      ...             ...      ...            ...
304      F   56        1               1        1              2
305      M   70        2               1        1              1
306      M   58        2               1        1              1
307      M   67        2               1        2              1
308      M   62        1               1        1              2

     CHRONIC DISEASE  FATIGUE  ALLERGY  WHEEZING  ALCOHOL CONSUMING  COUGH
ING  \
0                  1        2        1         2                  2
2
1                  2        2        2         1                  1
1
2                  1        2        1         2                  1
2
3                  1        1        1         1                  2
1
4                  1        1        1         2                  1
2
..               ...      ...      ...       ...                ...
...
304                2        2        1         1                  2
2
305                1        2        2         2                  2
2
306                1        1        2         2                  2
2
307                1        2        2         1                  2
2
308                1        2        2         2                  2
1

     SHORTNESS OF BREATH  SWALLOWING DIFFICULTY  CHEST PAIN
0                      2                       2           2
1                      2                       2           2
2                      2                       1           2
3                      1                       2           2
4                      2                       1           1
..                   ...                     ...         ...
304                    2                       2           1
305                    2                       1           2
306                    1                       1           2
307                    2                       1           2
308                    1                       2           1

[309 rows x 15 columns]
```

In [8]:
```
#independent_variable
y = lung_data. iloc[:,-1:]
print(y)
```

```
    LUNG_CANCER
0           YES
1           YES
2            NO
3            NO
4            NO
..          ...
304         YES
305         YES
306         YES
307         YES
308         YES

[309 rows x 1 columns]
```

In [9]:
```
lung_data.GENDER = lung_data.GENDER.map({"M":1,"F":2})
lung_data.LUNG_CANCER = lung_data.LUNG_CANCER.map({"YES":1,"NO":2})
```

In [10]:
```
lung_data.shape
```

Out[10]:
```
(309, 16)
```

In [11]:
```
lung_data.isnull().sum()
```

Out[11]:
```
GENDER                    0
AGE                       0
SMOKING                   0
YELLOW_FINGERS            0
ANXIETY                   0
PEER_PRESSURE             0
CHRONIC DISEASE           0
FATIGUE                   0
ALLERGY                   0
WHEEZING                  0
ALCOHOL CONSUMING         0
COUGHING                  0
SHORTNESS OF BREATH       0
SWALLOWING DIFFICULTY     0
CHEST PAIN                0
LUNG_CANCER               0
dtype: int64
```

In [12]:
```
lung_data.dtypes
```

Out[12]:
```
GENDER                  int64
AGE                     int64
SMOKING                 int64
YELLOW_FINGERS          int64
ANXIETY                 int64
PEER_PRESSURE           int64
CHRONIC DISEASE         int64
FATIGUE                 int64
ALLERGY                 int64
WHEEZING                int64
ALCOHOL CONSUMING       int64
COUGHING                int64
SHORTNESS OF BREATH     int64
SWALLOWING DIFFICULTY   int64
CHEST PAIN              int64
LUNG_CANCER             int64
dtype: object
```

In [13]:
```
lung_data.head()
```

Out[13]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FA
0	1	69	1	2	2	1	1	
1	1	74	2	1	1	1	2	
2	2	59	1	1	1	2	1	
3	1	63	2	2	2	1	1	
4	2	63	1	2	1	1	1	

In [14]:
```
lung_data.tail()
```

Out[14]:

SSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	S
2	2	2	1	1	2	2	2	
1	1	2	2	2	2	2	2	
1	1	1	2	2	2	2	1	
1	1	2	2	1	2	2	2	
2	1	2	2	2	2	1	1	

In [15]:
```
#the describe() method returns description of data in DataFrame
lung_data.describe()
```

Out[15]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE
count	309.000000	309.000000	309.000000	309.000000	309.000000	309.000000
mean	1.475728	62.673139	1.563107	1.569579	1.498382	1.501618
std	0.500221	8.210301	0.496806	0.495938	0.500808	0.500808
min	1.000000	21.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	57.000000	1.000000	1.000000	1.000000	1.000000
50%	1.000000	62.000000	2.000000	2.000000	1.000000	2.000000
75%	2.000000	69.000000	2.000000	2.000000	2.000000	2.000000
max	2.000000	87.000000	2.000000	2.000000	2.000000	2.000000

In [16]:
```
#the info() method prints information of the database
lung_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   GENDER                 309 non-null    int64
 1   AGE                    309 non-null    int64
 2   SMOKING                309 non-null    int64
 3   YELLOW_FINGERS         309 non-null    int64
 4   ANXIETY                309 non-null    int64
 5   PEER_PRESSURE          309 non-null    int64
 6   CHRONIC DISEASE        309 non-null    int64
 7   FATIGUE                309 non-null    int64
 8   ALLERGY                309 non-null    int64
 9   WHEEZING               309 non-null    int64
 10  ALCOHOL CONSUMING      309 non-null    int64
 11  COUGHING               309 non-null    int64
 12  SHORTNESS OF BREATH    309 non-null    int64
 13  SWALLOWING DIFFICULTY  309 non-null    int64
 14  CHEST PAIN             309 non-null    int64
 15  LUNG_CANCER            309 non-null    int64
dtypes: int64(16)
memory usage: 38.8 KB
```

In [17]:
```
#Splitting the Dataset: Training and Testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=1/3,random_s
```

In [18]:
```
lung_data['LUNG_CANCER'].value_counts()
```

Out[18]:
```
1    270
2     39
Name: LUNG_CANCER, dtype: int64
```

In [19]:
```
len(lung_data)
```

Out[19]: 309

In [20]:
```
len(x_test)
```

Out[20]: 103

In [21]:
```
len(x_train)
```

Out[21]: 206

In [22]:
```
#dependent_variable
x = lung_data.iloc[:,0:-1]
x
```

Out[22]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE
0	1	69	1	2	2	1	1
1	1	74	2	1	1	1	2
2	2	59	1	1	1	2	1
3	1	63	2	2	2	1	1
4	2	63	1	2	1	1	1
...	...	...	...	...	...	...	...
304	2	56	1	1	1	2	2
305	1	70	2	1	1	1	1
306	1	58	2	1	1	1	1
307	1	67	2	1	2	1	1
308	1	62	1	1	1	2	1

309 rows × 15 columns

In [23]:
```
#independent_variable
y = lung_data.iloc[:,-1:]
y
```

Out[23]:

	LUNG_CANCER
0	1
1	1
2	2
3	2
4	2
...	...
304	1
305	1
306	1
307	1
308	1

309 rows × 1 columns

In [24]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [26]:
```
from sklearn.linear_model import LogisticRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=1/3,random_s
```

Logistic Regression

In [27]:
```
#Fitting simple linear regression to the training test
Model1 = LogisticRegression()
Model1.fit(x_train, y_train)
#Predicting the test set results
prediction1 = Model1.predict(x_test)
```

In [28]:
```
prediction1
```

Out[28]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1,
       1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [29]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
confusion_matrix(y_test,prediction1)
```

Out[29]:
```
array([[85,  2],
       [10,  6]], dtype=int64)
```

In [30]:
```
accuracy_score(y_test,prediction1)
```

Out[30]:
```
0.883495145631068
```

In [31]:
```
from sklearn.metrics import precision_score
probs = Model1.predict_proba(x_test)
precision_score(y_test, prediction1, average = None)
```

Out[31]:
```
array([0.89473684, 0.75      ])
```

In [32]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

# assuming your predicted and actual labels are stored in variables y_pred 
accuracy = accuracy_score(y_test, prediction1)
precision = precision_score(y_test, prediction1)
recall = recall_score(y_test, prediction1)
f1 = f1_score(y_test, prediction1)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 0.883495145631068
Precision: 0.8947368421052632
Recall: 0.9770114942528736
F1 score: 0.9340659340659342
```

In [33]:
```
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

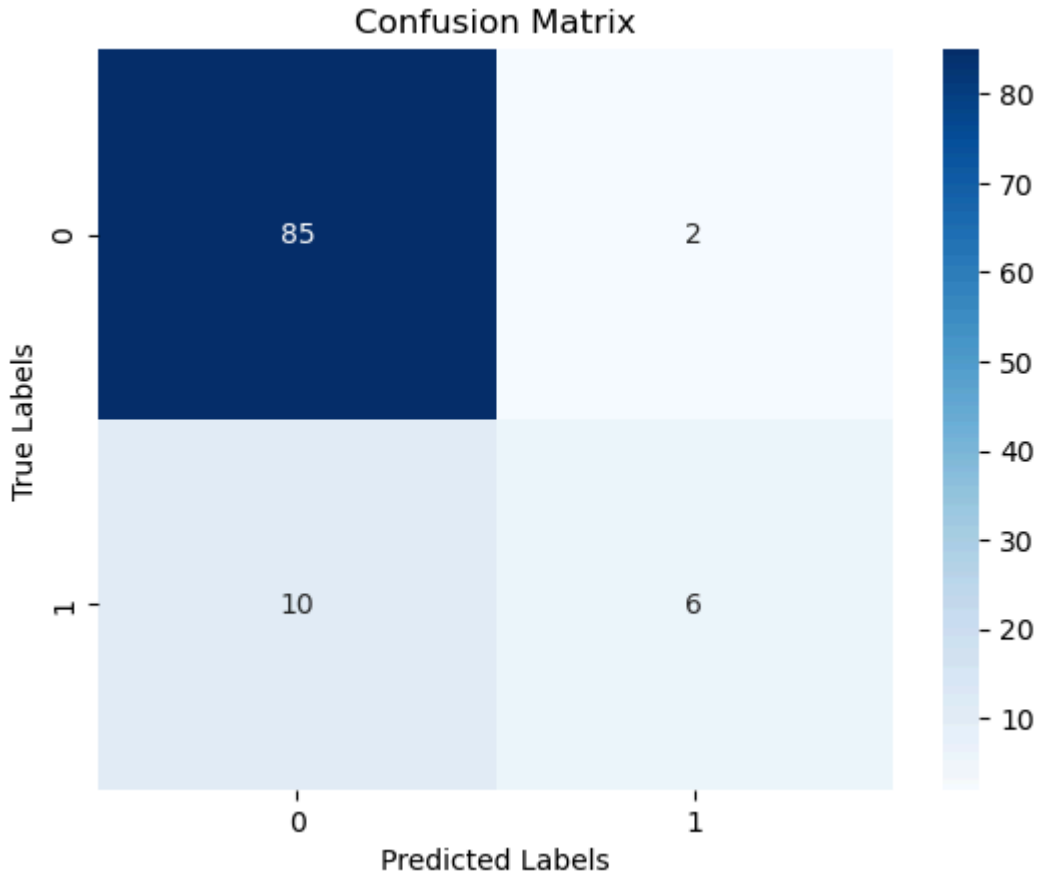
In [34]:
```
recall_score(y_test, prediction1, average = None)
```

Out[34]:
```
array([0.97701149, 0.375     ])
```

In [35]:
```
f1_score(y_test, prediction1, average = None)
```

Out[35]:
```
array([0.93406593, 0.5       ])
```

In [36]:
```
cm = confusion_matrix(y_true = y_test, y_pred = prediction1)
#plot_confusion_matrix(cm,level,title = "confusion_matrix")
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



KNN

In [40]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [41]:
```
#Fitting K-NN to the Training set
classifier = KNeighborsClassifier(n_neighbors = 3, metric = "minkowski", p
classifier.fit(x_train, y_train)
```

Out[41]:
```
KNeighborsClassifier(n_neighbors=3)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [42]:
```
#Predicting the Test set result
prediction2 = classifier.predict(x_test)
```

In [43]:
```
prediction2
```

Out[43]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [44]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
confusion_matrix(y_test,prediction2)
```

Out[44]:
```
array([[86,  1],
       [12,  4]], dtype=int64)
```

In [45]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

# assuming your predicted and actual labels are stored in variables y_pred (
accuracy = accuracy_score(y_test, prediction2)
precision = precision_score(y_test, prediction2)
recall = recall_score(y_test, prediction2)
f1 = f1_score(y_test, prediction2)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 0.8737864077669902
Precision: 0.8775510204081632
Recall: 0.9885057471264368
F1 score: 0.9297297297297297
```

In [46]:
```
accuracy_score(y_test,prediction2)
```

Out[46]:
```
0.8737864077669902
```

In [47]:
```
probs = Model1.predict_proba(x_test)
precision_score(y_test, prediction2, average = None)
```

Out[47]:
```
array([0.87755102, 0.8       ])
```

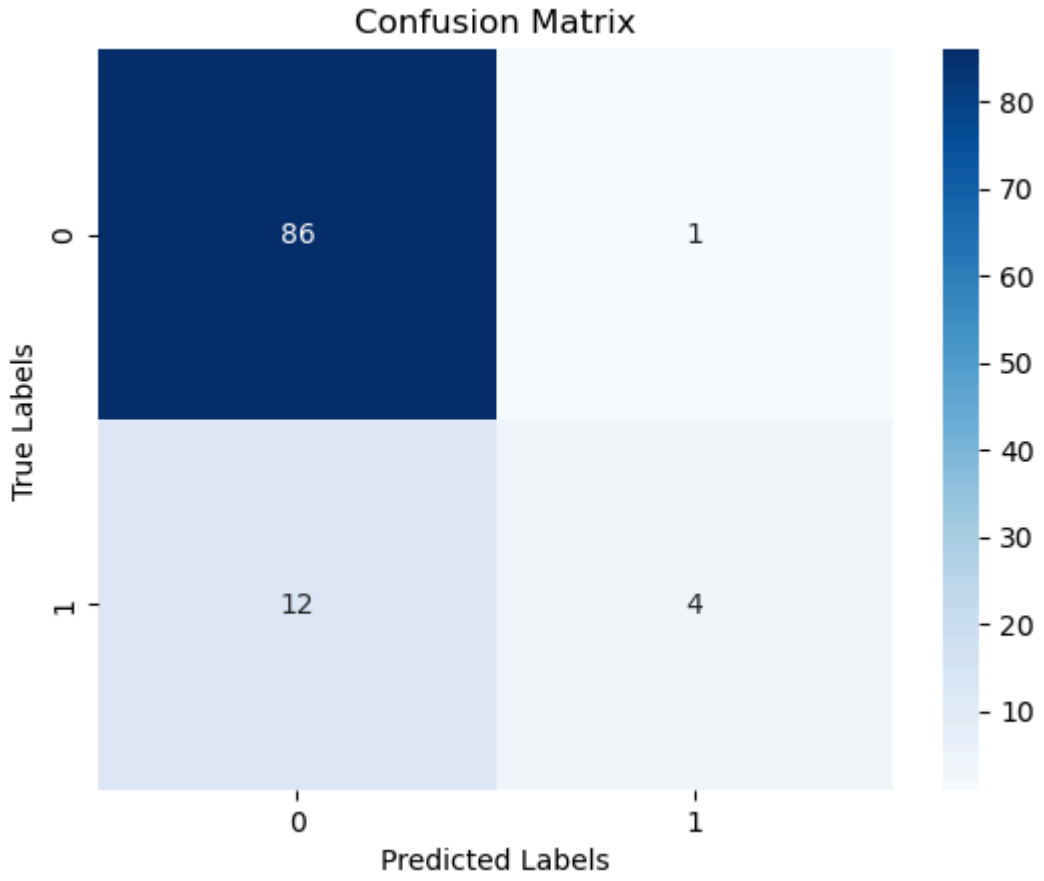
In [48]:
```
recall_score(y_test, prediction2, average = None)
```

Out[48]:
```
array([0.98850575, 0.25      ])
```

In [49]:
```
f1_score(y_test, prediction2, average = None)
```

Out[49]:
```
array([0.92972973, 0.38095238])
```

In [50]:
```
cm = confusion_matrix(y_true = y_test, y_pred = prediction2)
#plot_confusion_matrix(cm,level,title = "confusion_matrix")
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



Decision Tree

In [51]:
```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state = 0,criterion = "entropy")
tree.fit(x_train, y_train)
prediction3 = classifier.predict(x_test)
```

In [52]:
```
prediction3
```

Out[52]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [53]:
```
confusion_matrix(y_test,prediction3)
```

Out[53]:
```
array([[86,  1],
       [12,  4]], dtype=int64)
```

In [54]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

# assuming your predicted and actual labels are stored in variables y_pred
accuracy = accuracy_score(y_test, prediction3)
precision = precision_score(y_test, prediction3)
recall = recall_score(y_test, prediction3)
f1 = f1_score(y_test, prediction3)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 0.8737864077669902
Precision: 0.8775510204081632
Recall: 0.9885057471264368
F1 score: 0.9297297297297297
```

In [55]:
```
accuracy_score(y_test,prediction3)
```

Out[55]:
```
0.8737864077669902
```

In [56]:
```
probs = Model1.predict_proba(x_test)
precision_score(y_test, prediction3, average = None)
```

Out[56]:
```
array([0.87755102, 0.8       ])
```

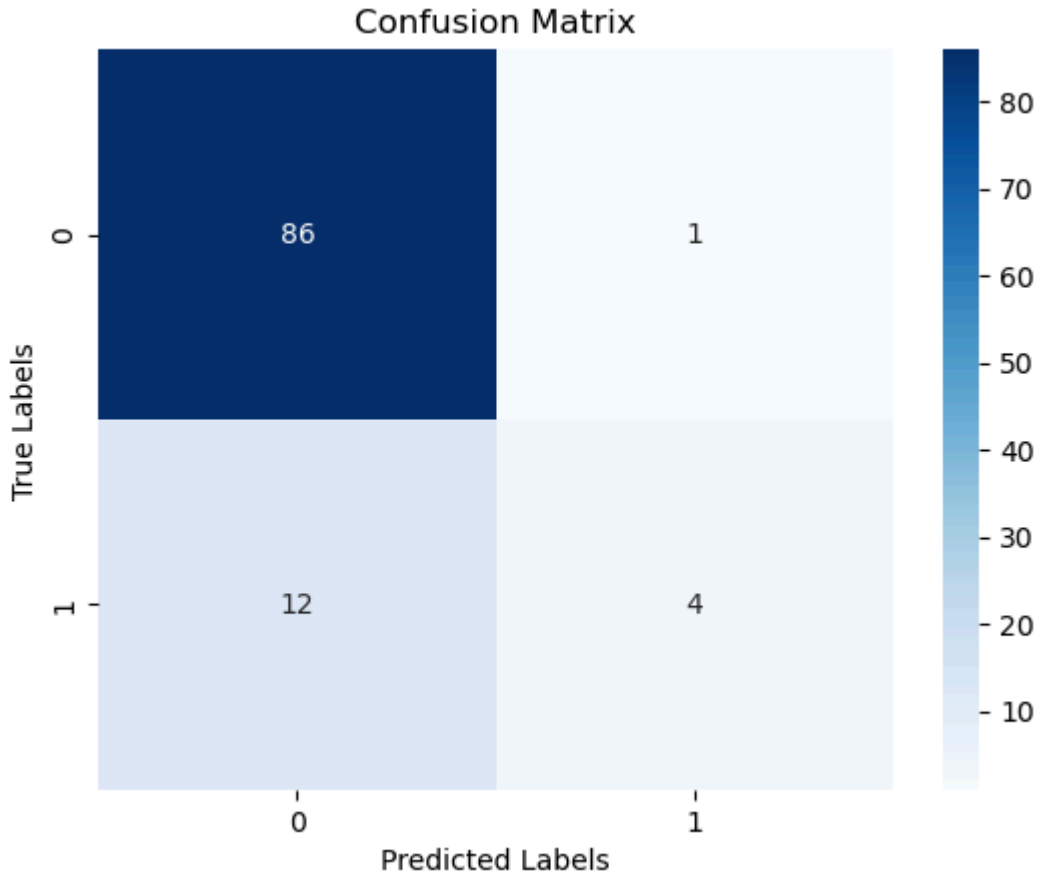
In [57]:
```
recall_score(y_test, prediction3, average = None)
```

Out[57]:
```
array([0.98850575, 0.25      ])
```

In [58]:
```
f1_score(y_test, prediction3, average = None)
```

Out[58]:
```
array([0.92972973, 0.38095238])
```

In [59]:
```
cm = confusion_matrix(y_true = y_test, y_pred = prediction3)
#plot_confusion_matrix(cm,level,title = "confusion_matrix")
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



Support Vector Machine

In [60]:
```
#Support Vector Machine
from sklearn.ensemble import BaggingClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
svm = OneVsRestClassifier(BaggingClassifier(SVC(C=10,kernel='rbf',random_st
svm.fit(x_train, y_train)
prediction4 = svm.predict(x_test)
```

In [61]:
```
confusion_matrix(y_test,prediction4)
```

Out[61]:
```
array([[87,  0],
       [16,  0]], dtype=int64)
```

In [62]:
```
trics import precision_score, recall_score, f1_score

 predicted and actual labels are stored in variables y_pred and y_true, resp
racy_score(y_test, prediction4)
cision_score(y_test, prediction4)
_score(y_test, prediction4)
_test, prediction4)

:", accuracy)
n:", precision)
, recall)
:", f1)
```

```
Accuracy: 0.8446601941747572
Precision: 0.8446601941747572
Recall: 1.0
F1 score: 0.9157894736842105
```

In [63]:
```
accuracy_score(y_test,prediction4)
```

Out[63]:
```
0.8446601941747572
```

In [64]:
```
probs = Model1.predict_proba(x_test)
precision_score(y_test, prediction4, average = None)
```

Out[64]:
```
array([0.84466019, 0.        ])
```

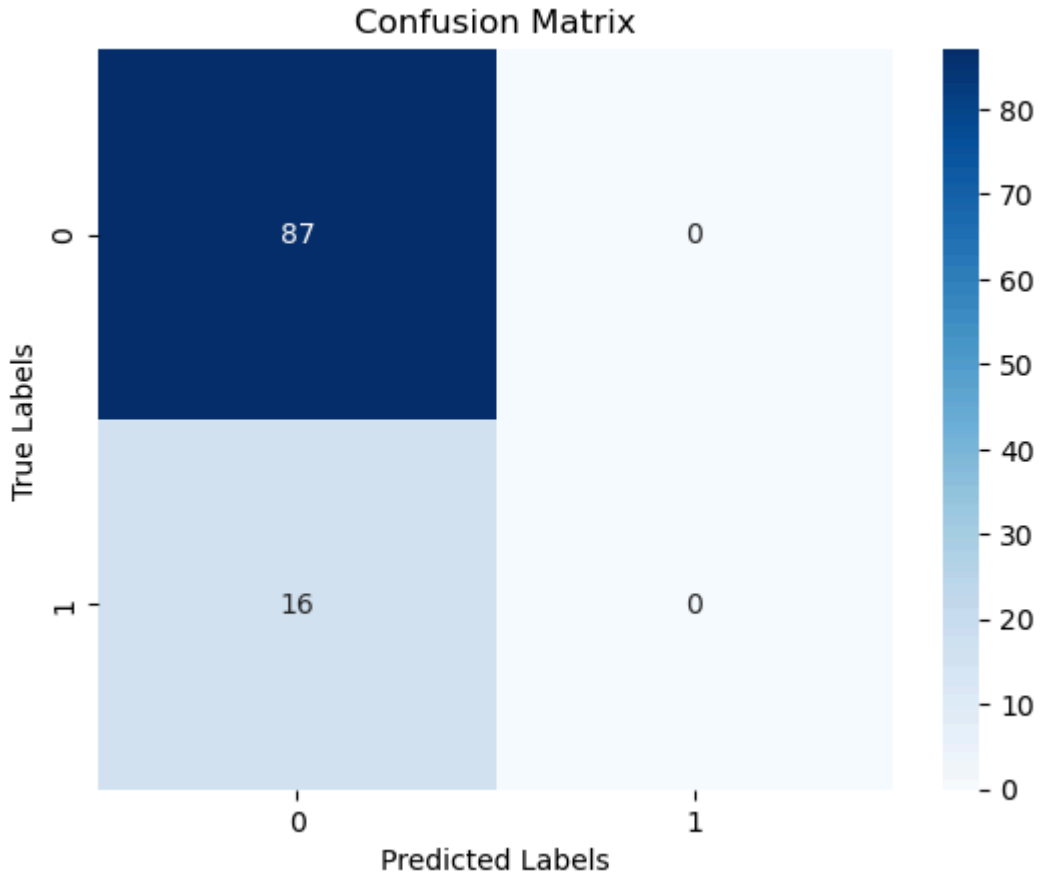
In [65]:
```
recall_score(y_test, prediction4, average = None)
```

Out[65]:
```
array([1., 0.])
```

In [66]:
```
f1_score(y_test, prediction4, average = None)
```

Out[66]:
```
array([0.91578947, 0.        ])
```

In [67]:
```
cm = confusion_matrix(y_true = y_test, y_pred = prediction4)
#plot_confusion_matrix(cm,level,title = "confusion_matrix")
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



Navie Bayes

In [68]:
```
from sklearn.naive_bayes import GaussianNB
nbcla = GaussianNB()
nbcla.fit(x_train, y_train)
prediction5 = nbcla.predict(x_test)
```

In [69]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
confusion_matrix(y_test,prediction5)
```

Out[69]:
```
array([[81,  6],
       [ 8,  8]], dtype=int64)
```

In [70]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

# assuming your predicted and actual labels are stored in variables y_pred (
accuracy = accuracy_score(y_test, prediction5)
precision = precision_score(y_test, prediction5)
recall = recall_score(y_test, prediction5)
f1 = f1_score(y_test, prediction5)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 0.8640776699029126
Precision: 0.9101123595505618
Recall: 0.9310344827586207
F1 score: 0.9204545454545454
```

In [71]:
```
accuracy_score(y_test,prediction5)
```

Out[71]:
```
0.8640776699029126
```

In [72]:
```
probs = Model1.predict_proba(x_test)
precision_score(y_test, prediction5, average = None)
```

Out[72]:
```
array([0.91011236, 0.57142857])
```

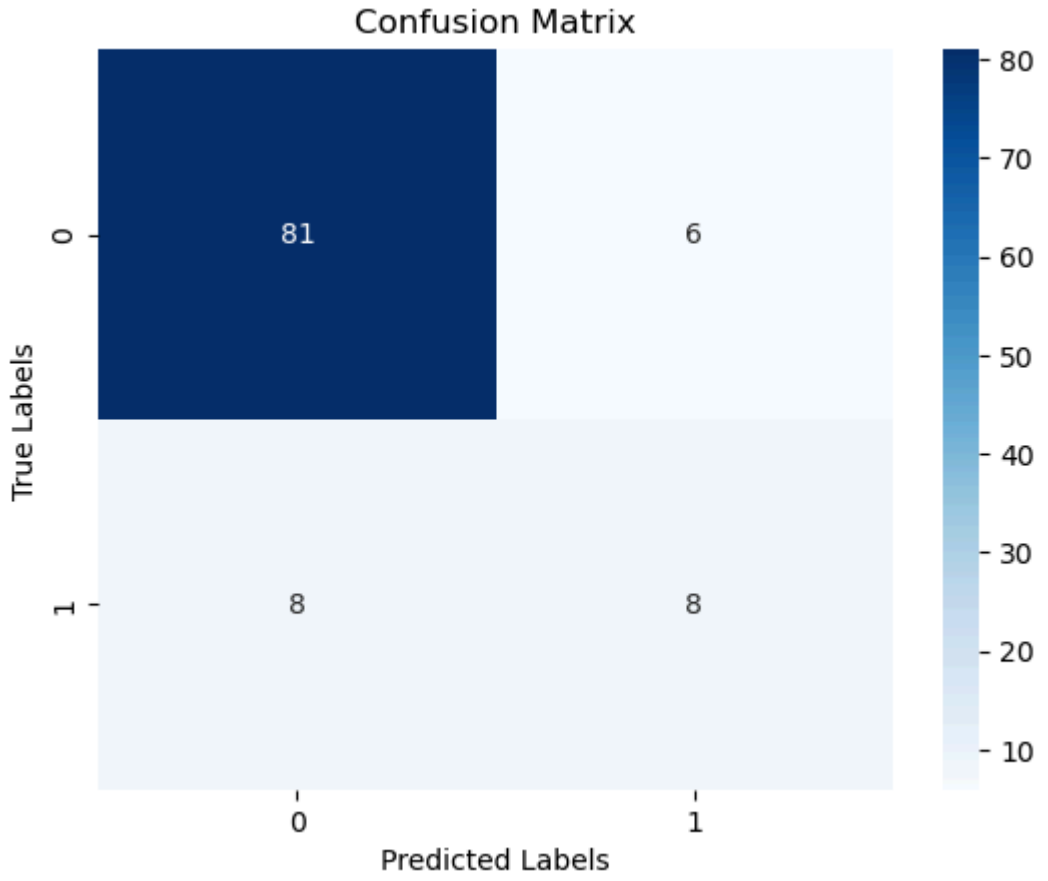
In [73]:
```
recall_score(y_test, prediction5, average = None)
```

Out[73]:
```
array([0.93103448, 0.5       ])
```

In [74]:
```
f1_score(y_test, prediction5, average = None)
```

Out[74]:
```
array([0.92045455, 0.53333333])
```

In [75]:
```
cm = confusion_matrix(y_true = y_test, y_pred = prediction5)
#plot_confusion_matrix(cm,level,title = "confusion_matrix")
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



Random Forest

In [76]:
```
from sklearn.ensemble import RandomForestClassifier

# Initialize the classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model using training dataset
rf_classifier.fit(x_train, y_train)

# Make predictions on test dataset
prediction6 = rf_classifier.predict(x_test)

# Evaluate the accuracy of the model
#accuracy = rf_classifier.score(x_test, y_test)
#print("Accuracy:", accuracy)
```

In [77]:
```
confusion_matrix(y_test,prediction6)
```

Out[77]:
```
array([[85,  2],
       [ 9,  7]], dtype=int64)
```

In [78]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

# assuming your predicted and actual labels are stored in variables y_pred (
accuracy = accuracy_score(y_test, prediction6)
precision = precision_score(y_test, prediction6)
recall = recall_score(y_test, prediction6)
f1 = f1_score(y_test, prediction6)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 0.8932038834951457
Precision: 0.9042553191489362
Recall: 0.9770114942528736
F1 score: 0.9392265193370166
```

In [79]:
```
accuracy_score(y_test,prediction6)
```

Out[79]:
```
0.8932038834951457
```

In [80]:
```
probs = Model1.predict_proba(x_test)
precision_score(y_test, prediction6, average = None)
```

Out[80]:
```
array([0.90425532, 0.77777778])
```

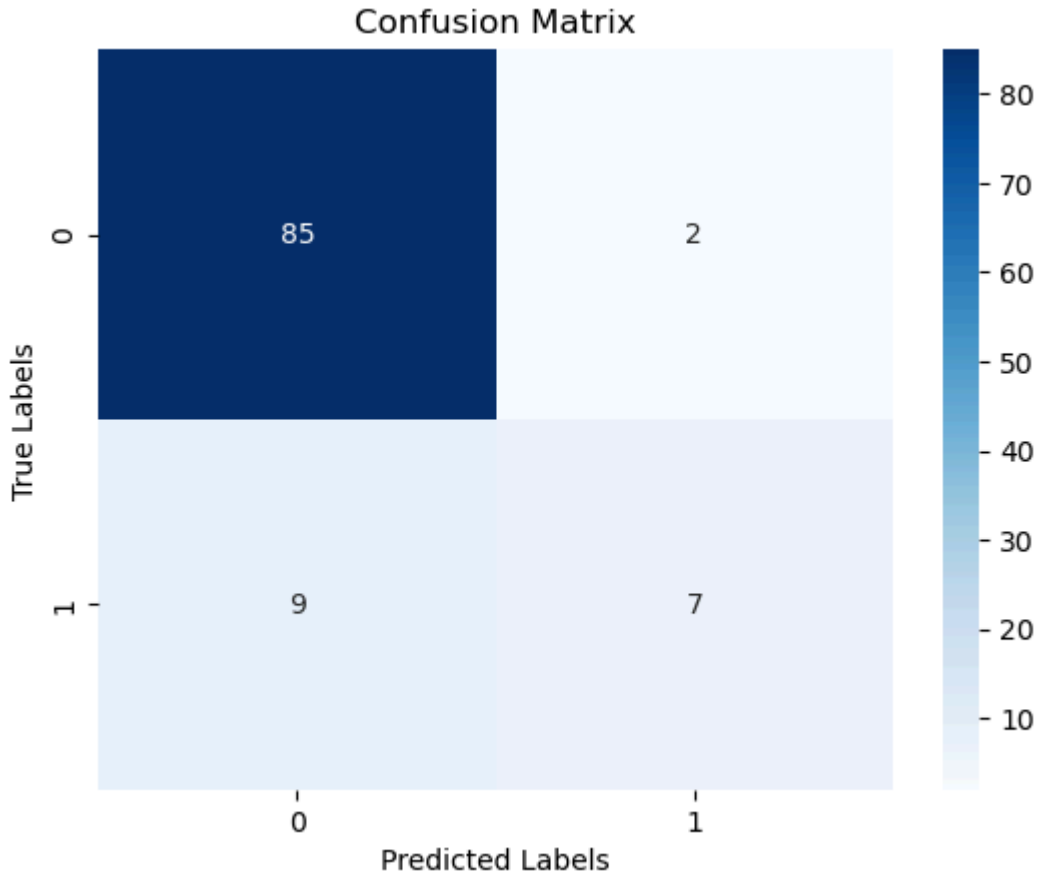
In [81]:
```
recall_score(y_test, prediction6, average = None)
```

Out[81]:
```
array([0.97701149, 0.4375    ])
```

In [82]:
```
f1_score(y_test, prediction6, average = None)
```

Out[82]:
```
array([0.93922652, 0.56      ])
```

In [83]:
```
cm = confusion_matrix(y_true = y_test, y_pred = prediction6)
#plot_confusion_matrix(cm,level,title = "confusion_matrix")
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

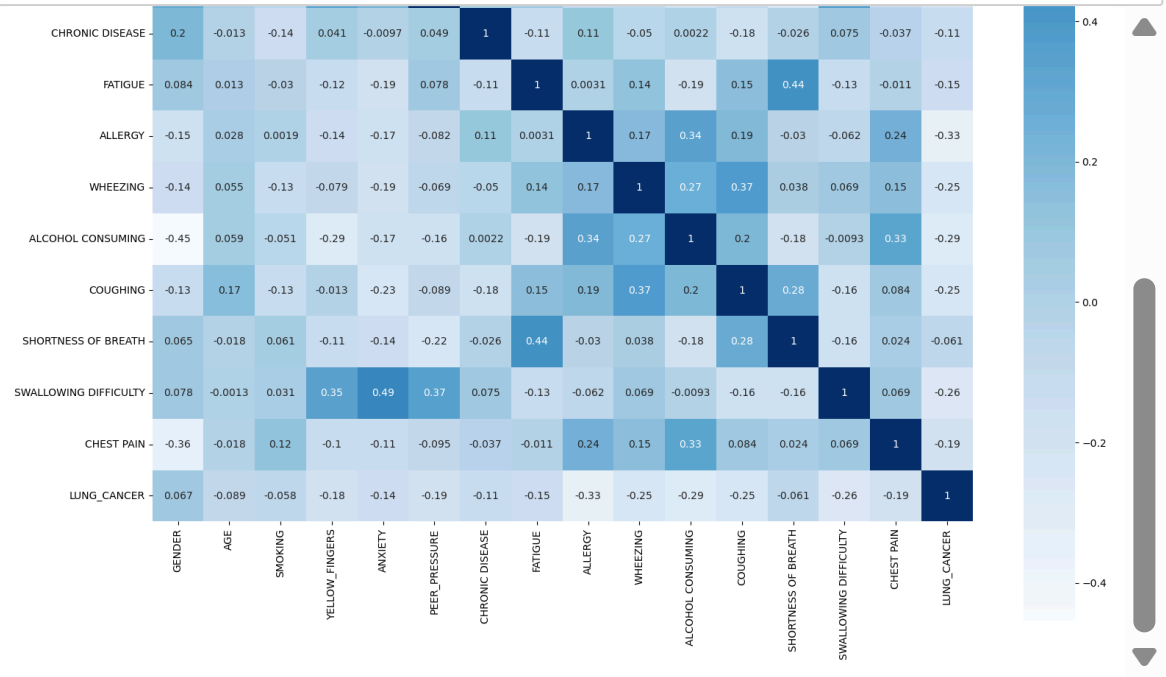


In [84]:
```
#Finding Correlation
cn=lung_data.corr()
cn
```

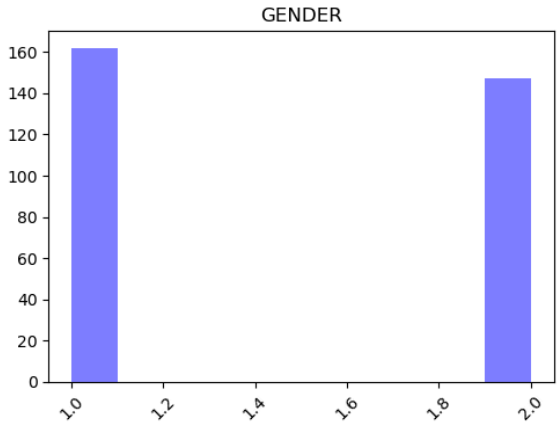
Out[84]:

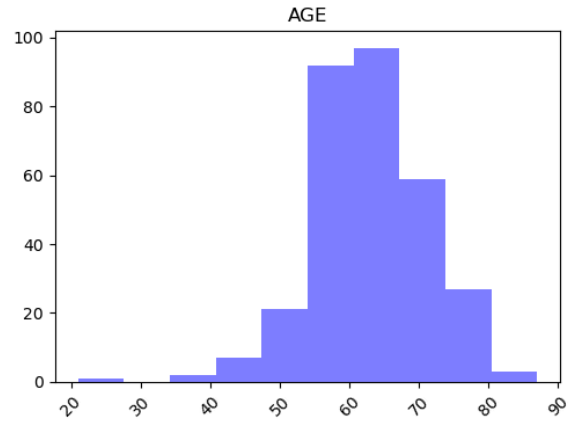
	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRI
GENDER	1.000000	-0.021306	-0.036277	0.212959	0.152127	(
AGE	-0.021306	1.000000	-0.084475	0.005205	0.053170	(
SMOKING	-0.036277	-0.084475	1.000000	-0.014585	0.160267	-(
YELLOW_FINGERS	0.212959	0.005205	-0.014585	1.000000	0.565829	(
ANXIETY	0.152127	0.053170	0.160267	0.565829	1.000000	(
PEER_PRESSURE	0.275564	0.018685	-0.042822	0.323083	0.216841	1
CHRONIC DISEASE	0.204606	-0.012642	-0.141522	0.041122	-0.009678	(
FATIGUE	0.083560	0.012614	-0.029575	-0.118058	-0.188538	(
ALLERGY	-0.154251	0.027990	0.001913	-0.144300	-0.165750	-(
WHEEZING	-0.141207	0.055011	-0.129426	-0.078515	-0.191807	-(
ALCOHOL CONSUMING	-0.454268	0.058985	-0.050623	-0.289025	-0.165750	-(
COUGHING	-0.133303	0.169950	-0.129471	-0.012640	-0.225644	-(
SHORTNESS OF BREATH	0.064911	-0.017513	0.061264	-0.105944	-0.144077	-(
SWALLOWING DIFFICULTY	0.078161	-0.001270	0.030718	0.345904	0.489403	(
CHEST PAIN	-0.362958	-0.018104	0.120117	-0.104829	-0.113634	-(
LUNG_CANCER	0.067254	-0.089465	-0.058179	-0.181339	-0.144947	-(

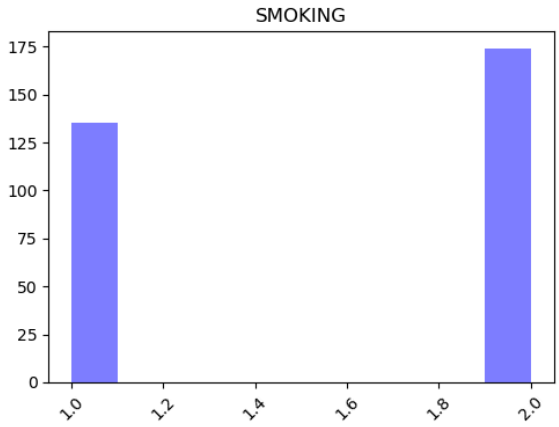
In [85]:
```
#Correlation
cmap=sns.diverging_palette(260,-10,s=50, l=75, n=6,
as_cmap=True)
plt.subplots(figsize=(18,18))
sns.heatmap(cn,cmap="Blues",annot=True, square=True)
plt.show()
```

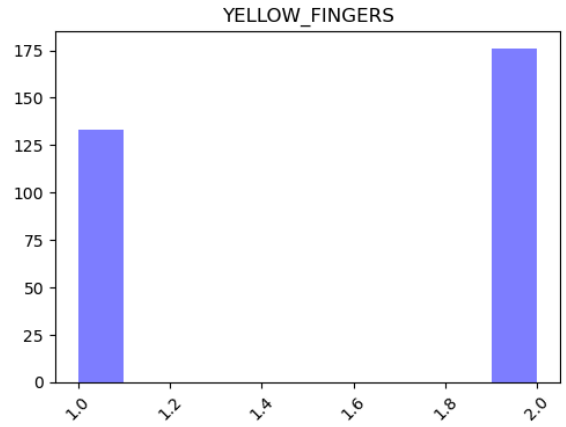


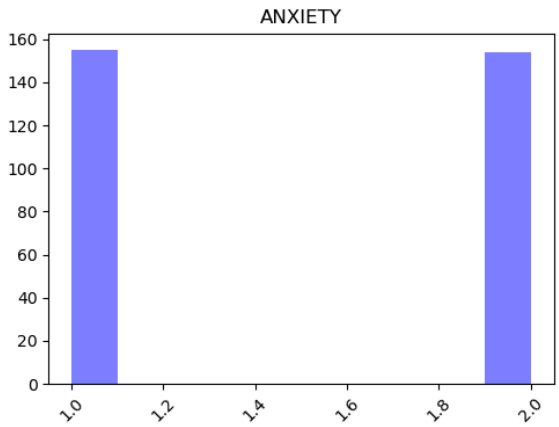
In [86]:
```
num_list = list(lung_data.columns)

fig = plt.figure(figsize=(10,30))

for i in range(len(num_list)):
    plt.subplot(8,2,i+1)
    plt.title(num_list[i])
    plt.xticks(rotation=45)
    plt.hist(lung_data[num_list[i]],color='blue',alpha=0.5)

plt.tight_layout()
```

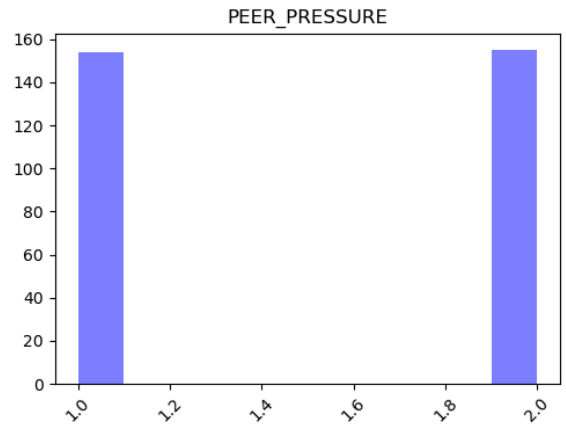

GENDER
160
140
120
100
80
60
40
20
0
1.0
1.2
1.4
1.6
1.8
2.0

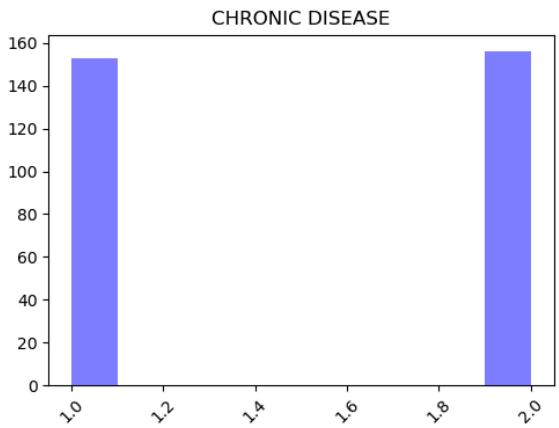

AGE
100
80
60
40
20
0
20
30
40
50
60
70
80
90

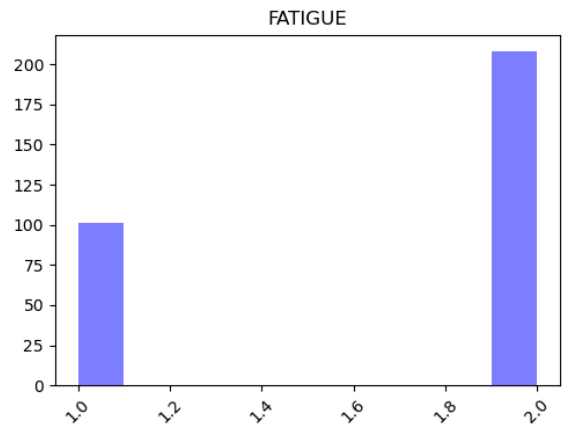

SMOKING
175
150
125
100
75
50
25
0
1.0
1.2
1.4
1.6
1.8
2.0

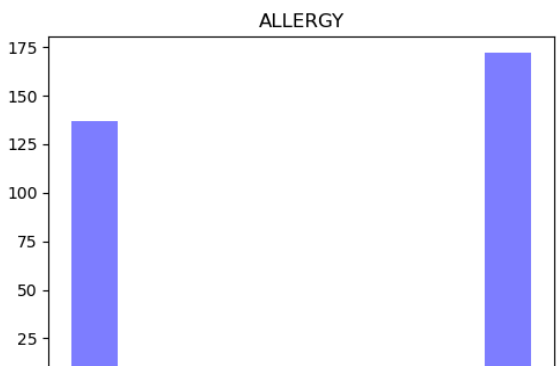

YELLOW_FINGERS
175
150
125
100
75
50
25
0
1.0
1.2
1.4
1.6
1.8
2.0

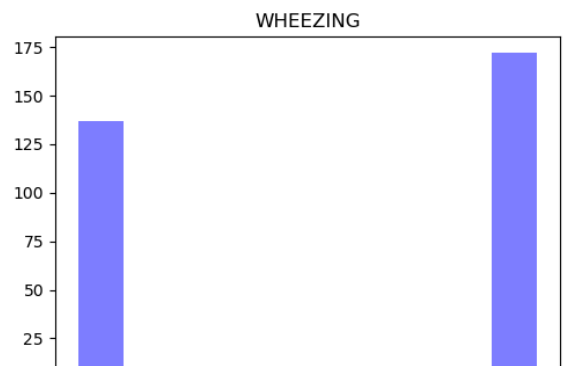

ANXIETY
160
140
120
100
80
60
40
20
0
1.0
1.2
1.4
1.6
1.8
2.0

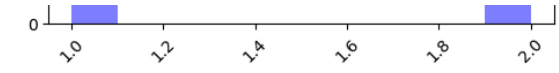

PEER_PRESSURE
160
140
120
100
80
60
40
20
0
1.0
1.2
1.4
1.6
1.8
2.0

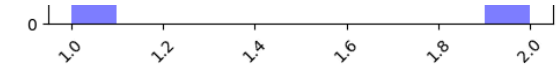

CHRONIC DISEASE
160
140
120
100
80
60
40
20
0
1.0
1.2
1.4
1.6
1.8
2.0

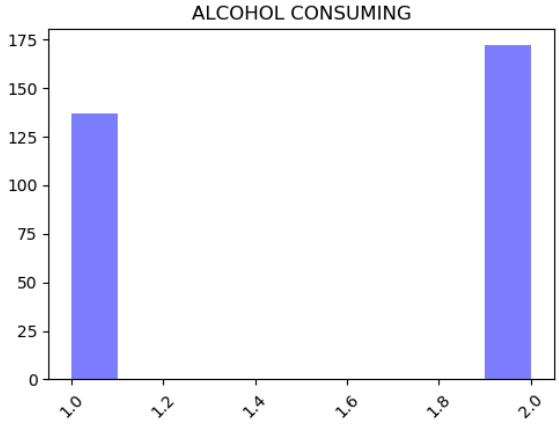

FATIGUE
200
175
150
125
100
75
50
25
0
1.0
1.2
1.4
1.6
1.8
2.0

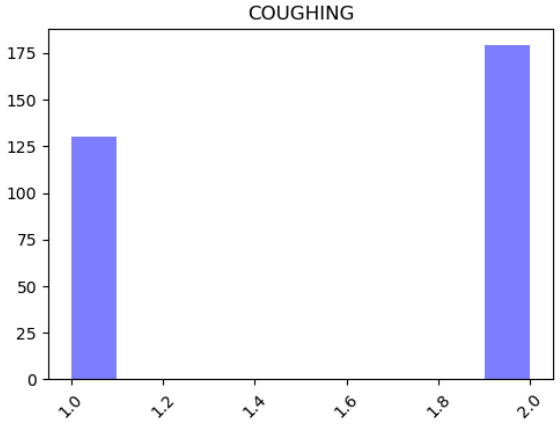

ALLERGY
175
150
125
100
75
50
25

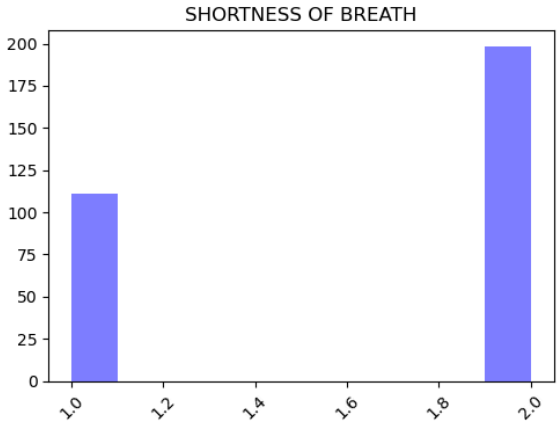

WHEEZING
175
150
125
100
75
50
25

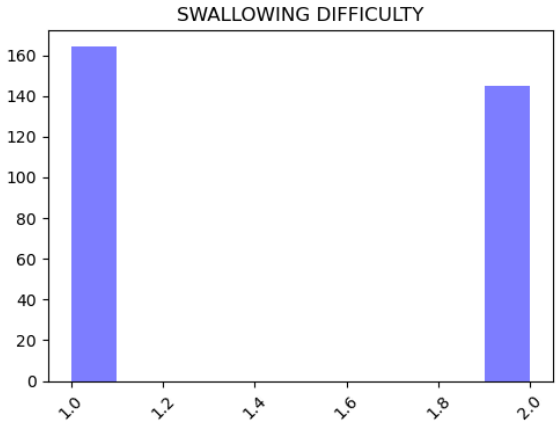

0
1.0
1.2
1.4
1.6
1.8
2.0

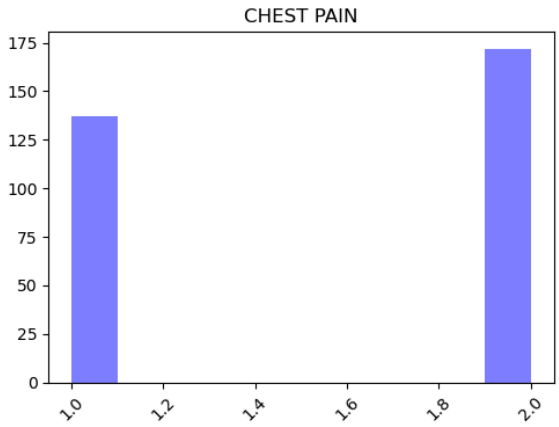

0
1.0
1.2
1.4
1.6
1.8
2.0

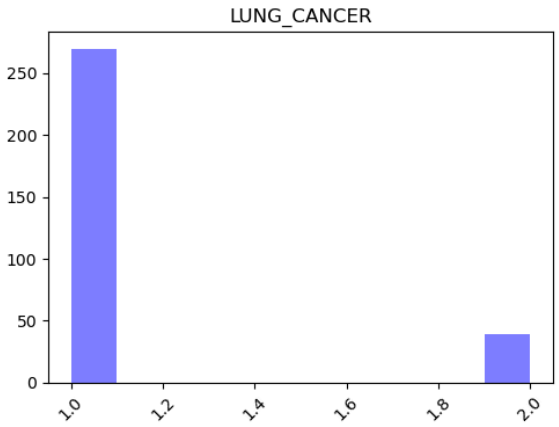

ALCOHOL CONSUMING
175
150
125
100
75
50
25
0
1.0
1.2
1.4
1.6
1.8
2.0


COUGHING
175
150
125
100
75
50
25
0
1.0
1.2
1.4
1.6
1.8
2.0


SHORTNESS OF BREATH
200
175
150
125
100
75
50
25
0
1.0
1.2
1.4
1.6
1.8
2.0


SWALLOWING DIFFICULTY
160
140
120
100
80
60
40
20
0
1.0
1.2
1.4
1.6
1.8
2.0


CHEST PAIN
175
150
125
100
75
50
25
0
1.0
1.2
1.4
1.6
1.8
2.0


LUNG_CANCER
250
200
150
100
50
0
1.0
1.2
1.4
1.6
1.8
2.0