Abstrature

- In this project, I want to visualize the variables of COVID-19 and use regression model to predict the death of COVID-19.

Dataset Information

This dataset contains an enormous number of anonymized patient-related information including pre-conditions. The raw dataset consists of 21 unique features and 1,048,576 unique patients. In the Boolean features, 1 means "yes" and 2 means "no". values as 97 and 99 are missing data.

- sex: female or male.
- age: of the patient.
- classification: covid test findings. Values 1-3 mean that the patient was diagnosed with covid in different degrees. 4 or higher means that the patient is not a carrier of covid or that the test is inconclusive.
- patient type: hospitalized or not hospitalized.
- pneumonia: whether the patient already have air sacs inflammation or not.
- pregnancy: whether the patient is pregnant or not.
- diabetes: whether the patient has diabetes or not.
- copd: Indicates whether the patient has Chronic obstructive pulmonary disease or not.
- asthma: whether the patient has asthma or not.
- inmsupr: whether the patient is immunosuppressed or not.
- hypertension: whether the patient has hypertension or not.
- cardiovascular: whether the patient has heart or blood vessels related disease.
- renal chronic: whether the patient has chronic renal disease or not.
- other disease: whether the patient has other disease or not.
- obesity: whether the patient is obese or not.
- tobacco: whether the patient is a tobacco user.
- usmr: Indicates whether the patient treated medical units of the first, second or third level.
- medical unit: type of institution of the National Health System that provided the care.
- intubed: whether the patient was connected to the ventilator.# breath machine
- icu: Indicates whether the patient had been admitted to an Intensive Care Unit.
- death: indicates whether the patient died or recovered.

Work Flow:

1. Data Cleaning

1) Import dataset

2) Data Preparing & Cleaning

3) Missing Value Analysis

2. Data visualisation

1) Death Distribution

2) Age Distribution

3) Age-Death

4) Age-Death-Sex

5) Sex-Death

6) Obesity-Death

7) Correlation Between Features

3. Regression Model

1) Variables Selection

2) Train Test Split

3) Logistic Regression

4) Undersampling

5) Train Test Split After Undersampling

6) Logistic Regression After Undersampling

7) Logistic Regression Curve

4. Conlusions

1.Data Cleaning

1) Import dataset

In [1]:
```
# import modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv("/Users/zhouwenguang/Desktop/semester_1/data_visualisatio
df.head()
```

Out[2]:

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE
0	2	1	1	1	03/05/2020	97	1	65
1	2	1	2	1	03/06/2020	97	1	72
2	2	1	2	2	09/06/2020	1	2	55
3	2	1	1	1	12/06/2020	97	2	53
4	2	1	2	1	21/06/2020	97	2	68

5 rows × 21 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 21 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   USMER                 1048575 non-null  int64
 1   MEDICAL_UNIT          1048575 non-null  int64
 2   SEX                   1048575 non-null  int64
 3   PATIENT_TYPE          1048575 non-null  int64
 4   DATE_DIED             1048575 non-null  object
 5   INTUBED               1048575 non-null  int64
 6   PNEUMONIA             1048575 non-null  int64
 7   AGE                   1048575 non-null  int64
 8   PREGNANT              1048575 non-null  int64
 9   DIABETES              1048575 non-null  int64
 10  COPD                  1048575 non-null  int64
 11  ASTHMA                1048575 non-null  int64
 12  INMSUPR               1048575 non-null  int64
 13  HIPERTENSION          1048575 non-null  int64
 14  OTHER_DISEASE         1048575 non-null  int64
 15  CARDIOVASCULAR        1048575 non-null  int64
 16  OBESITY               1048575 non-null  int64
 17  RENAL_CHRONIC         1048575 non-null  int64
 18  TOBACCO               1048575 non-null  int64
 19  CLASIFFICATION_FINAL  1048575 non-null  int64
 20  ICU                   1048575 non-null  int64
dtypes: int64(20), object(1)
memory usage: 168.0+ MB
```

In [4]:
```
df.isna().sum()
```

Out[4]:
```
USMER                   0
MEDICAL_UNIT            0
SEX                     0
PATIENT_TYPE            0
DATE_DIED               0
INTUBED                 0
PNEUMONIA               0
AGE                     0
PREGNANT                0
DIABETES                0
COPD                    0
ASTHMA                  0
INMSUPR                 0
HIPERTENSION            0
OTHER_DISEASE           0
CARDIOVASCULAR          0
OBESITY                 0
RENAL_CHRONIC           0
TOBACCO                 0
CLASIFFICATION_FINAL    0
ICU                     0
dtype: int64
```

observations

- We have no NaN values but we will have missing values.

In [5]:
```
# observe the values of each variables
for i in df.columns:
    print(i,len(df[i].unique()))
```

```
USMER 2
MEDICAL_UNIT 13
SEX 2
PATIENT_TYPE 2
DATE_DIED 401
INTUBED 4
PNEUMONIA 3
AGE 121
PREGNANT 4
DIABETES 3
COPD 3
ASTHMA 3
INMSUPR 3
HIPERTENSION 3
OTHER_DISEASE 3
CARDIOVASCULAR 3
OBESITY 3
RENAL_CHRONIC 3
TOBACCO 3
CLASIFFICATION_FINAL 7
ICU 4
```

In [6]:
```
# patients died or not
df.DATE_DIED.value_counts()
```

Out[6]:
```
9999-99-99    971633
06/07/2020      1000
07/07/2020       996
13/07/2020       990
16/06/2020       979
                 ...
24/11/2020         1
17/12/2020         1
08/12/2020         1
16/03/2021         1
22/04/2021         1
Name: DATE_DIED, Length: 401, dtype: int64
```

In [7]:
```
df.PNEUMONIA.value_counts()
```

Out[7]:
```
2     892534
1     140038
99     16003
Name: PNEUMONIA, dtype: int64
```

observations

- We have some features that we expect them to have just 2 unique values but we see that these features have 3 or 4 unique values. For example the feature "PNEUMONIA" has 3 unique values (1,2,99) 99 represents NaN values. Hence we will just take the rows that includes 1 and 2 values.
- In "DATE_DIED" column, we have 971633 "9999-99-99" values which represent alive patients so i will take this feature as a "DEATH" that includes wether the patient died or not.

2) Data Preparing & Cleaning

Dealing with missing values

In [8]:
```
df = df[(df.PNEUMONIA == 1) | (df.PNEUMONIA == 2)]
df = df[(df.DIABETES == 1) | (df.DIABETES == 2)]
df = df[(df.COPD == 1) | (df.COPD == 2)]
df = df[(df.ASTHMA == 1) | (df.ASTHMA == 2)]
df = df[(df.INMSUPR == 1) | (df.INMSUPR == 2)]
df = df[(df.HIPERTENSION == 1) | (df.HIPERTENSION == 2)]
df = df[(df.OTHER_DISEASE == 1) | (df.OTHER_DISEASE == 2)]
df = df[(df.CARDIOVASCULAR == 1) | (df.CARDIOVASCULAR == 2)]
df = df[(df.OBESITY == 1) | (df.OBESITY == 2)]
df = df[(df.RENAL_CHRONIC == 1) | (df.RENAL_CHRONIC == 2)]
df = df[(df.TOBACCO == 1) | (df.TOBACCO == 2)]
```

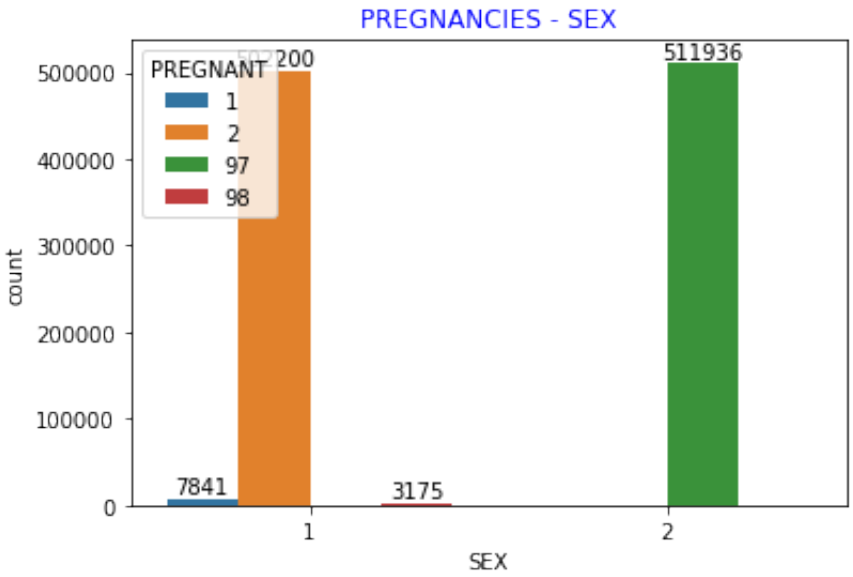
Changing the values of 'DATE_DIED'

In [9]:
```
# If we have "9999-99-99" values that means this patient is alive.

df["DEATH"] = [2 if each=="9999-99-99" else 1 for each in df.DATE_DIED]
```

Pregnant and Sex chart

In [10]:
```
plt.figure()
ax = sns.countplot(df.SEX, hue = df.PREGNANT)
for bars in ax.containers:
    ax.bar_label(bars)
plt.title("PREGNANCIES - SEX",color="blue")
```

Out[10]:
```
Text(0.5, 1.0, 'PREGNANCIES - SEX')
```

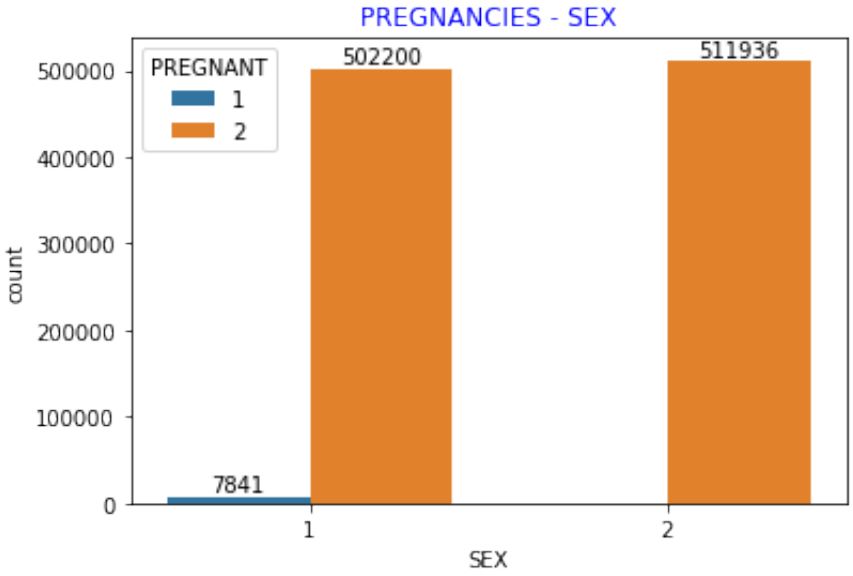


observations

- We see that all "97" values are for males and males can not be pregnant so we should convert 97 to 2.

In [11]:
```
# convert process according to observations above
df.PREGNANT = df.PREGNANT.replace(97,2)

# get rid of the missing values
df = df[(df.PREGNANT == 1) | (df.PREGNANT == 2)]
```

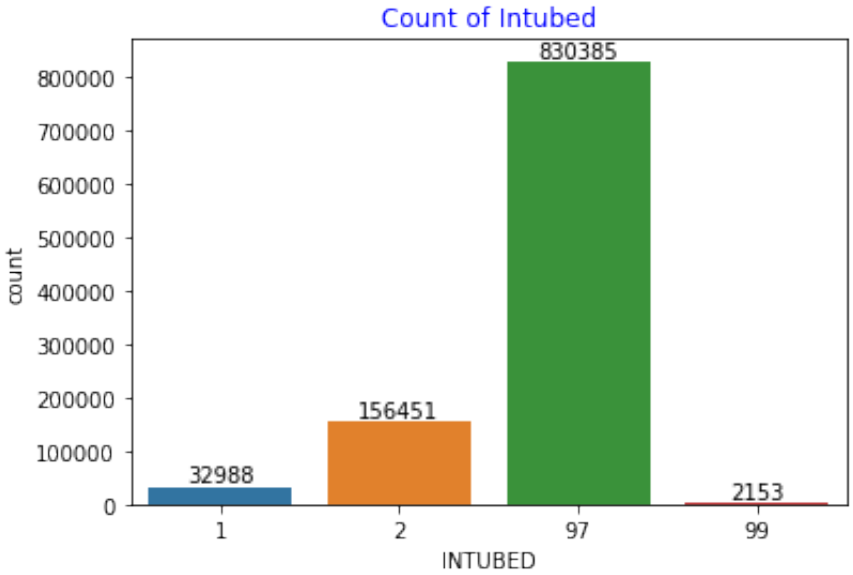
In [12]:
```
# after repalce missing values
plt.figure()
ax = sns.countplot(df.SEX, hue = df.PREGNANT)
for bars in ax.containers:
    ax.bar_label(bars)
plt.title("PREGNANCIES - SEX",color="blue")
plt.show()
```



3) Missing Value Analysis

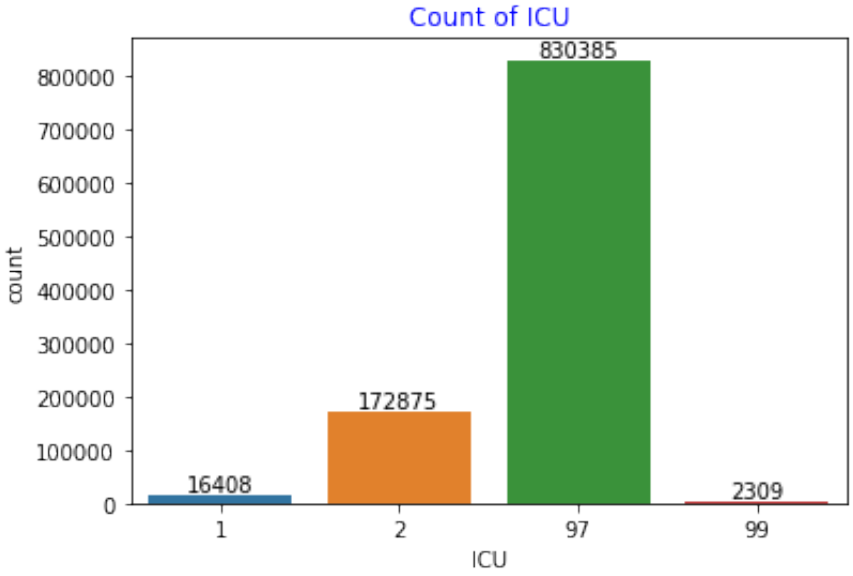
Missing value analysis of "INTUBED" feature

In [13]:
```
ax = sns.countplot(df.INTUBED)
plt.bar_label(ax.containers[0])
plt.title("Count of Intubed",color="blue")
plt.show()
```



Missing value analysis of "ICU" feature

In [14]:
```
ax = sns.countplot(df.ICU)
plt.bar_label(ax.containers[0])
plt.title("Count of ICU",color="blue")
plt.show()
```



Droping the columns

- In "INTUBED" and "ICU" features there are too many missing values so I will drop them. Also we don't need "DATE_DIED" column anymore because we used this feature as a "DEATH" feature.

In [15]:
```
df.drop(columns=["INTUBED","ICU","DATE_DIED"], inplace=True)
```

Number of unique values of each variables after dropping

In [16]:
```
for i in df.columns:
    print(i,"=>\t",len(df[i].unique()))
```

```
USMER =>          2
MEDICAL_UNIT =>   13
SEX =>    2
PATIENT_TYPE =>   2
PNEUMONIA =>      2
AGE =>    121
PREGNANT =>       2
DIABETES =>       2
COPD =>   2
ASTHMA =>         2
INMSUPR =>        2
HIPERTENSION =>   2
OTHER_DISEASE =>          2
CARDIOVASCULAR =>         2
OBESITY =>        2
RENAL_CHRONIC =>          2
TOBACCO =>        2
CLASIFFICATION_FINAL =>   7
DEATH =>          2
```

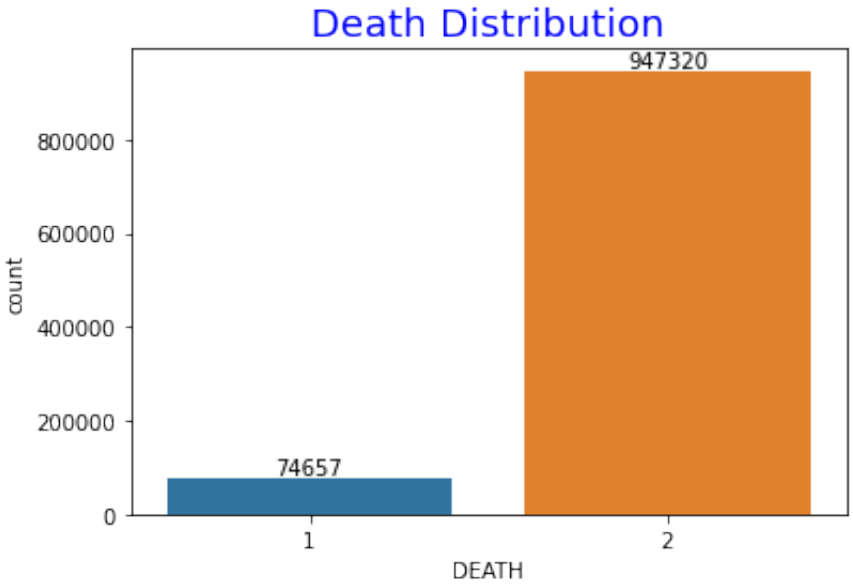
observations

- From the chart above, we can see that we have just one numeric variable which is called "AGE" the rest of them are categorical.

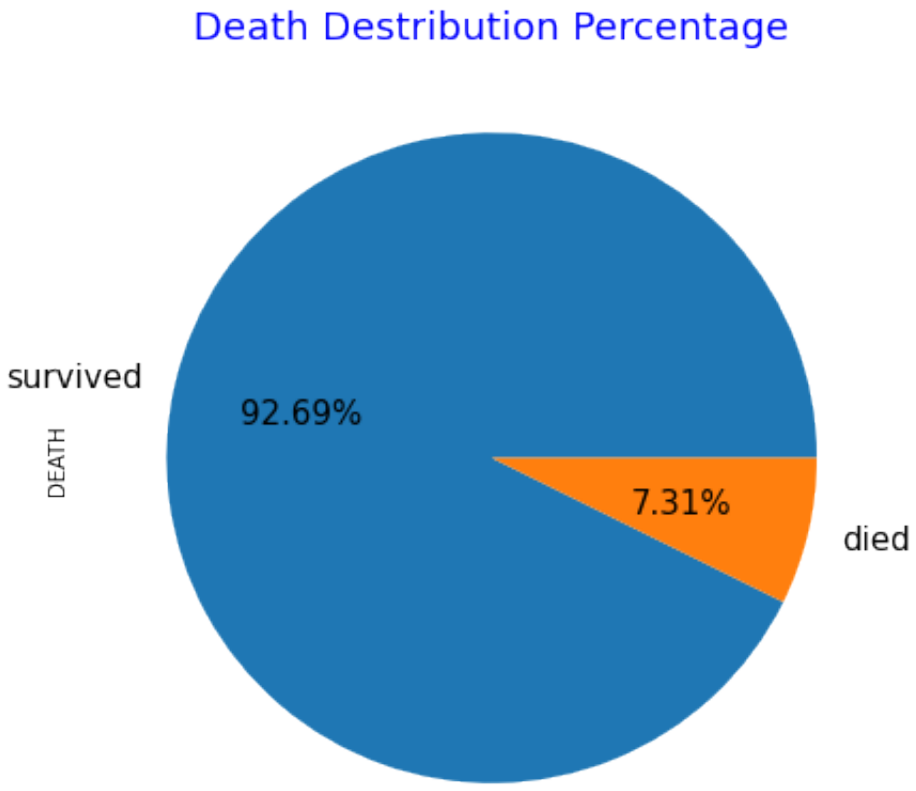
2.Data Visualization

1) Death Distribution

In [17]:
```
ax = sns.countplot(df.DEATH)
plt.bar_label(ax.containers[0])  # show the count label in the top of the
plt.title("Death Distribution", fontsize=18,color="blue");
```



In [18]:
```
plt.figure(figsize=[9,7])
df['DEATH'].value_counts().plot.pie(autopct='%.2f%%', labels = ('survived'
plt.title("Death Destribution Percentage", fontsize=18,color="blue")
plt.show()
```



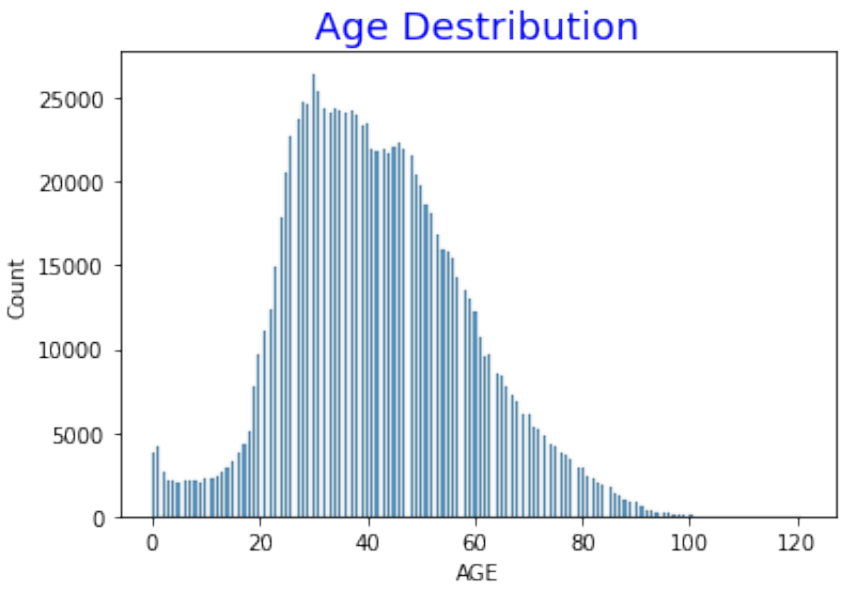
observations

- We can see that the values are distributed unbalanced in target column. This will probably cause an imbalance problem.

2) Age Distribution

In [19]:
```
sns.histplot(x=df.AGE)
plt.title("Age Destribution", color="blue", fontsize=18)
```

Out[19]:
```
Text(0.5, 1.0, 'Age Destribution')
```



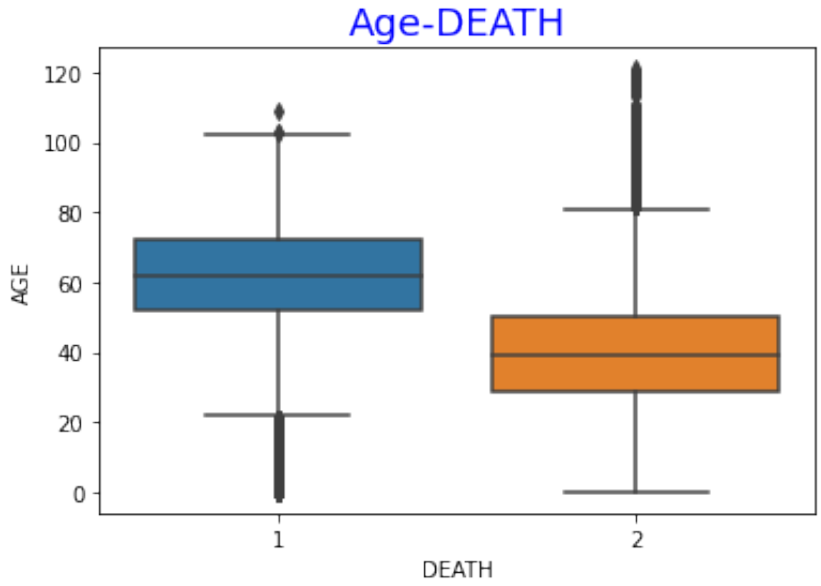
observations

- Patients are mainly between 20-60 years old.
- But patients between 1 and 2 years old are also susceptible to be infected.

3) Age-Death

In [20]:
```
sns.boxplot(x="DEATH", y="AGE",data=df)
plt.title("Age-DEATH",fontsize=18, color="blue")
```

Out[20]: Text(0.5, 1.0, 'Age-DEATH')



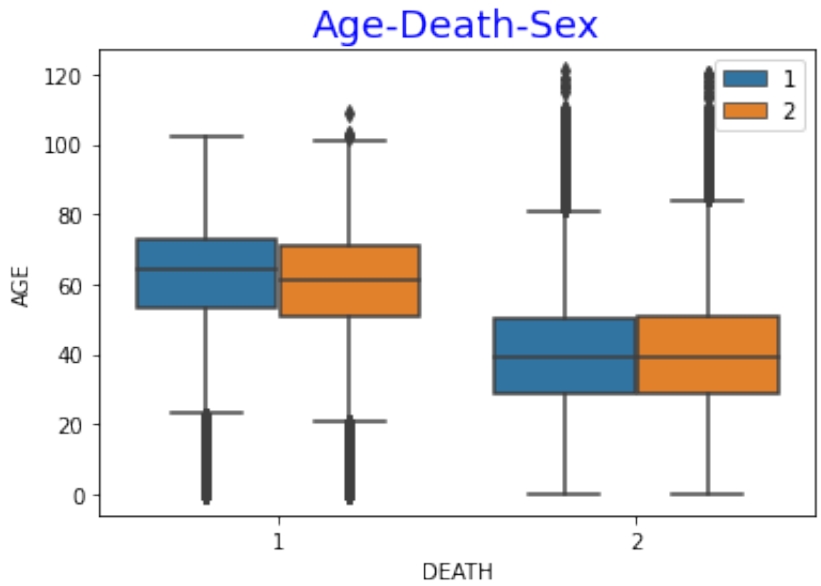
observations

- The older patients are more likely to die compare to younger patients.

4) Age-Death-Sex

In [21]:
```
sns.boxplot(x="DEATH", y="AGE",hue="SEX",data=df)
plt.title("Age-Death-Sex",fontsize=18, color="blue")
plt.legend(loc="best")
```

Out[21]: <matplotlib.legend.Legend at 0x7fda6d9393d0>

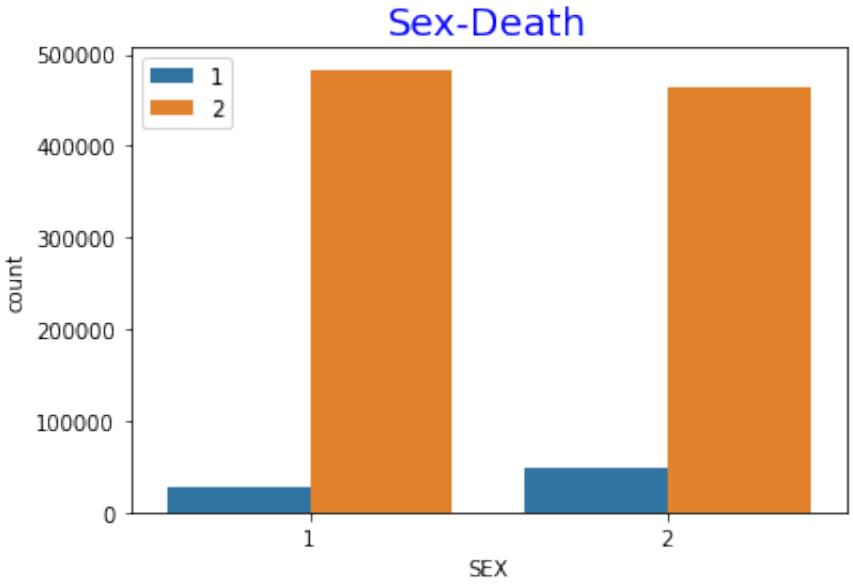


observations

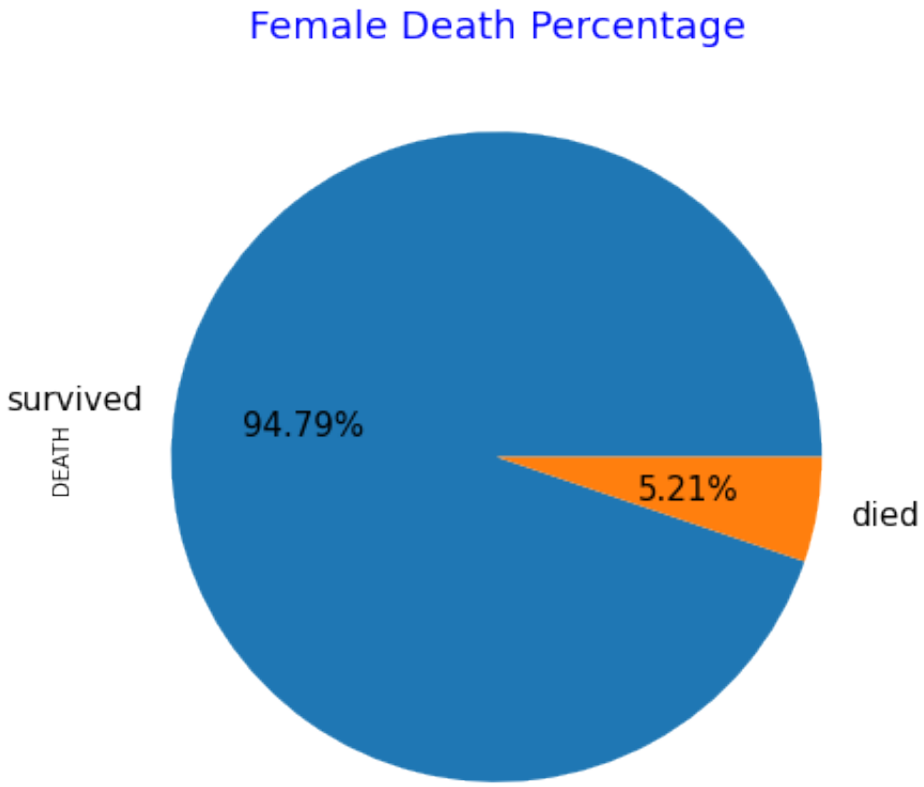
- There is no serious difference between males and females in terms of average rate of patients.

5) Sex-Death

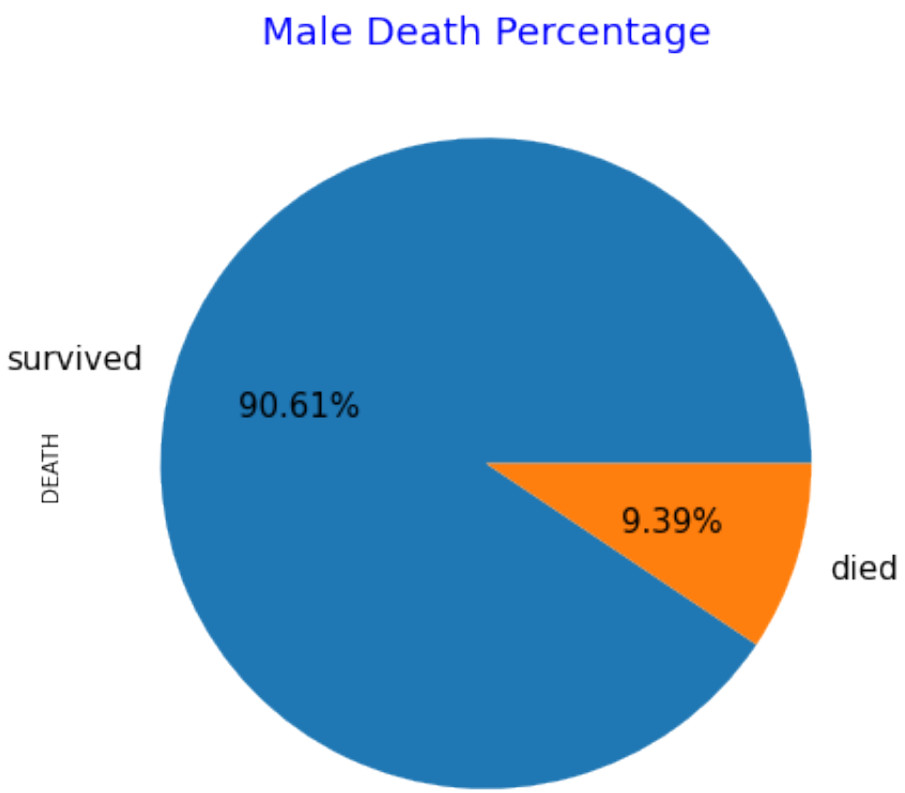
In [22]:
```
sns.countplot(df.SEX,hue=df.DEATH)
plt.title("Sex-Death",fontsize=18, color="blue")
plt.legend(loc="best")
plt.show()
```



In [23]:
```
female = df[df['SEX'] == 1]
plt.figure(figsize=[9,7])
female['DEATH'].value_counts().plot.pie(autopct='%.2f%%', labels = ('survi
plt.title("Female Death Percentage", fontsize=18,color="blue")
plt.show()
```



In [24]:
```
male = df[df['SEX'] == 2]
plt.figure(figsize=[9,7])
male['DEATH'].value_counts().plot.pie(autopct='%.2f%%', labels = ('survive
plt.title("Male Death Percentage", fontsize=18,color="blue")
plt.show()
```

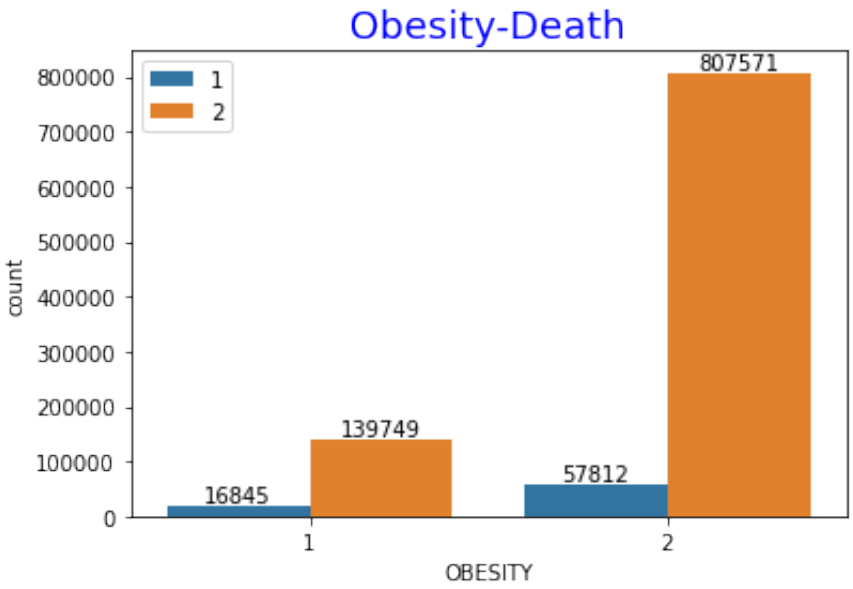


observations:

- Males are more likely to die of covid compare to females.

6) Obesity-Death

In [25]:
```
ax=sns.countplot(df.OBESITY,hue=df.DEATH)
plt.title("Obesity-Death",fontsize=18, color="blue")
plt.bar_label(ax.containers[0])
plt.bar_label(ax.containers[1])
plt.legend(loc="best")
plt.show()
```



In [26]:
```
print('obese patients death rate:',round(16845/(16845 + 139749),6))
print('non-obese patients death rate:', round(57812/(57812 + 807571),6))
```

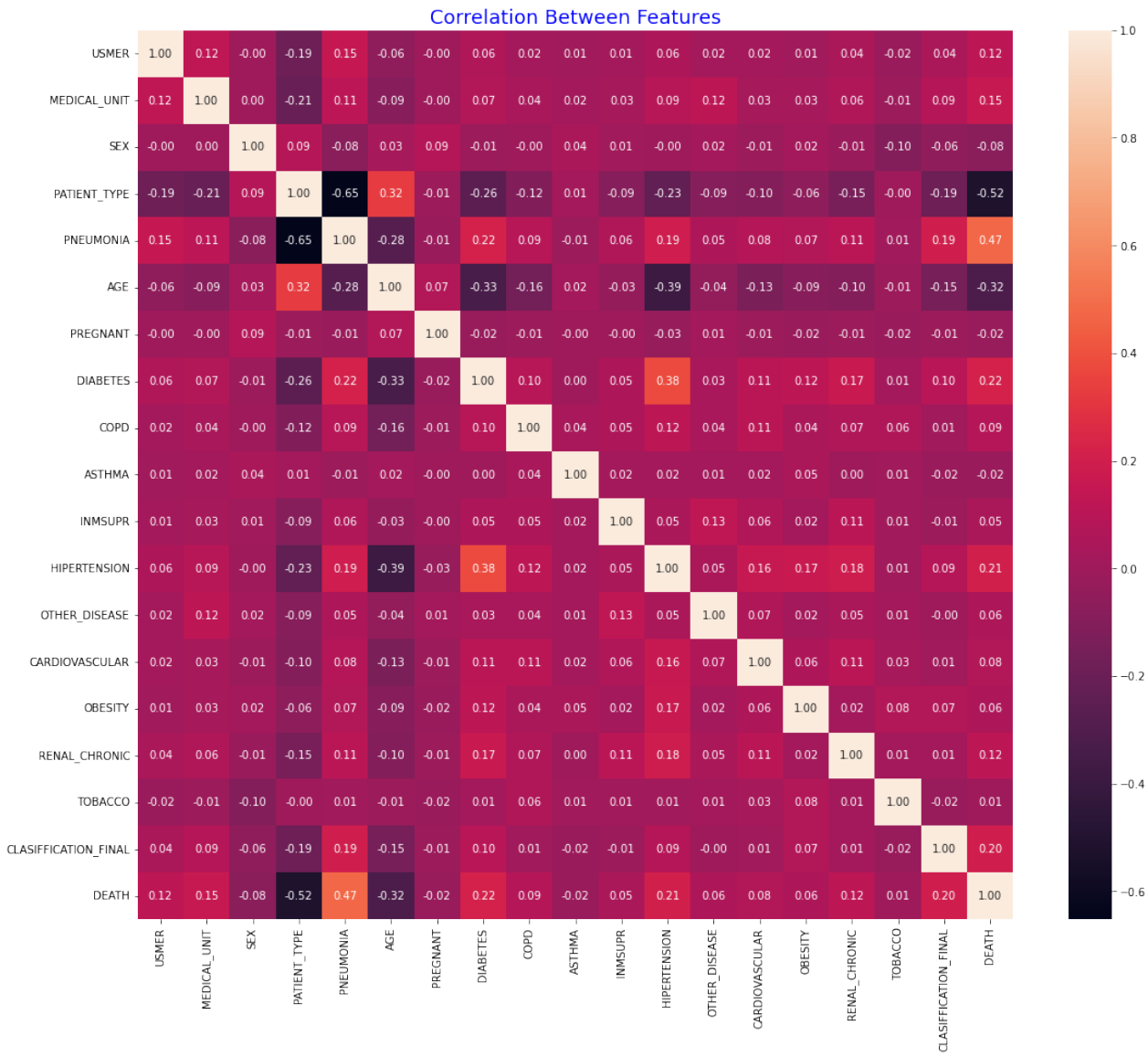
```
obese patients death rate: 0.107571
non-obese patients death rate: 0.066805
```

observations

- Obese patients are more likely to die from covid compare to non-obese patients.

7) Correlation Between Features

In [27]:
```
# Correlation Between Features
plt.figure(figsize=(18,15))
sns.heatmap(df.corr(),annot=True, fmt=".2f")
plt.title("Correlation Between Features",fontsize=18,color="blue");
```



observations

- PNEUMONIA variable has strongest positive correlation with DEATH variable.
- PATIENT_TYPE variable has strongest negative correlation with DEATH variable.

3. Regression Model

1) Variables Selection

- I will drop the variables that have low correlation(absolute value < 0.015) with "DEATH" variable.

In [28]:
```
unrelevant_columns = ["SEX","PREGNANT","COPD","ASTHMA","INMSUPR","OTHER_DIS
                      "OBESITY","TOBACCO","USMER","RENAL_CHRONIC"]

df.drop(columns=unrelevant_columns,inplace=True)
df.head()
```

Out[28]:

	MEDICAL_UNIT	PATIENT_TYPE	PNEUMONIA	AGE	DIABETES	HIPERTENSION	CLASIFFI
0	1	1	1	65	2	1	
1	1	1	1	72	2	1	
2	1	2	2	55	1	2	
3	1	1	2	53	2	2	
4	1	1	2	68	1	1	

dealing with the categorical variables which are not binary

In [29]:
```
df = pd.get_dummies(df,columns=["MEDICAL_UNIT","CLASIFFICATION_FINAL"],drop
df.head()
```

Out[29]:

	PATIENT_TYPE	PNEUMONIA	AGE	DIABETES	HIPERTENSION	DEATH	MEDICAL_UNIT_2
0	1	1	65	2	1	1	0
1	1	1	72	2	1	1	0
2	2	2	55	1	2	1	0
3	1	2	53	2	2	1	0
4	1	2	68	1	1	1	0

5 rows × 24 columns

Scaling the numeric feature

In [30]:
```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
df.AGE = scaler.fit_transform(df.AGE.values.reshape(-1,1))
df.head()
```

Out[30]:

	PATIENT_TYPE	PNEUMONIA	AGE	DIABETES	HIPERTENSION	DEATH	MEDICAL_UN
0	1	1	1.086957	2	1	1	
1	1	1	1.391304	2	1	1	
2	2	2	0.652174	1	2	1	
3	1	2	0.565217	2	2	1	
4	1	2	1.217391	1	1	1	

5 rows × 24 columns

Determining the "x" and "y"

In [31]:
```
x = df.drop(columns="DEATH")
y = df["DEATH"]
```

2) Train Test Split

In [32]:
```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(x,y, test_size=0.2, ra
print("Train_x :",train_x.shape)
print("Test_x :",test_x.shape)
print("Train_y :",train_y.shape)
print("Test_y :",test_y.shape)
```

```
Train_x : (817581, 23)
Test_x : (204396, 23)
Train_y : (817581,)
Test_y : (204396,)
```

3) Logistic Regression

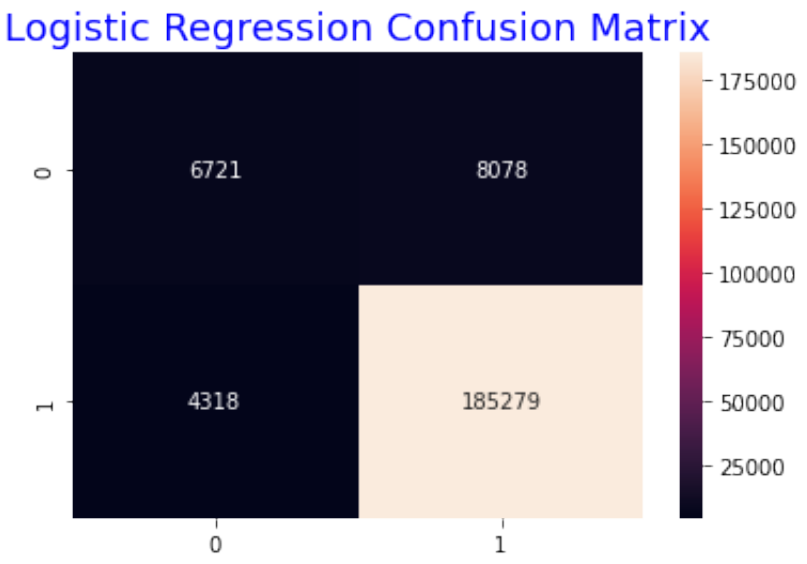
In [33]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(train_x,train_y)
print("Logistic Regression Accuracy :",logreg.score(test_x, test_y))
```

```
Logistic Regression Accuracy : 0.9393530206070569
```

In [34]:
```
from sklearn.metrics import f1_score

print("Logistic Regression F1 Score :",f1_score(test_y,logreg.predict(test_
```

```
Logistic Regression F1 Score : [0.5202415  0.96763058]
```

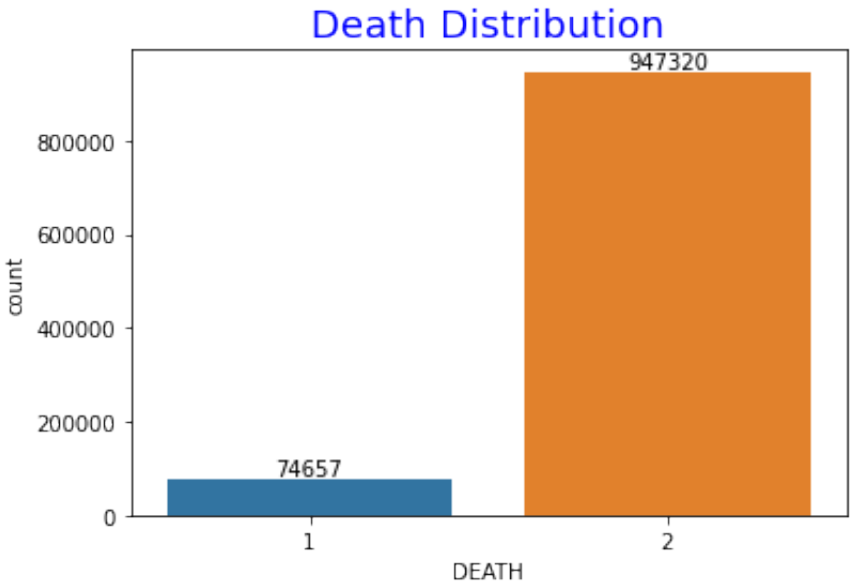
In [35]:
```
from sklearn.metrics import confusion_matrix

sns.heatmap(confusion_matrix(test_y, logreg.predict(test_x)), annot=True, 
plt.title("Logistic Regression Confusion Matrix",fontsize=18, color="blue"
```



Observations:

- We got well accuracy with Logistic Regression.
- But it can mislead us so we have to check the other metrics.
- When we look at the F1 Score it says that we predicted the patients who survived well but we can't say the same thing for dead patients.
- Also we see the same thing when we check the confusion matrix. This problem is because of imbalance dataset.

In [36]:
```
ax = sns.countplot(df.DEATH)
plt.bar_label(ax.containers[0])
plt.title("Death Distribution", fontsize=18,color="blue");
```

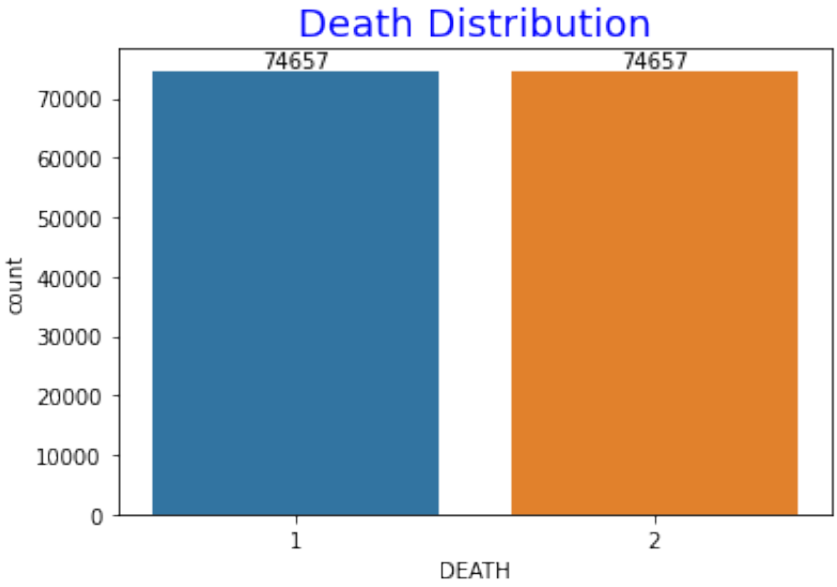


4) Undersampling

- Undersampling : Undersampling is a technique to balance uneven datasets by keeping all of the data in the minority class and decreasing the size of the majority class.
- If we use Oversampling our row number will increase so this is too many rows for computer.

In [37]:
```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=0)
x_resampled,y_resampled = rus.fit_resample(x,y)
```

In [38]:
```
ax = sns.countplot(y_resampled)
plt.bar_label(ax.containers[0])
plt.title("Death Distribution", fontsize=18,color="blue");
```



5) Train Test Split After Undersampling

In [39]:
```
train_x, test_x, train_y, test_y = train_test_split(x_resampled,y_resample
print("Train_x :",train_x.shape)
print("Test_x :",test_x.shape)
print("Train_y :",train_y.shape)
print("Test_y :",test_y.shape)
```

```
Train_x : (119451, 23)
Test_x : (29863, 23)
Train_y : (119451,)
Test_y : (29863,)
```

6) Logistic Regression After Undersampling

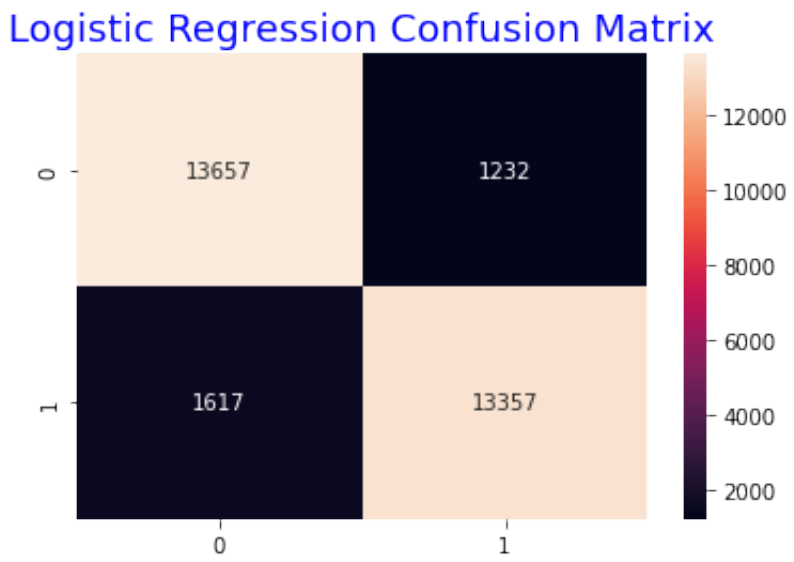
In [40]:
```
logreg = LogisticRegression()
logreg.fit(train_x,train_y)
print("Logistic Regression Accuracy :",logreg.score(test_x, test_y))
```

```
Logistic Regression Accuracy : 0.9045976626594783
```

In [41]:
```
print("Logistic Regression F1 Score :",
      f1_score(test_y,logreg.predict(test_x),average=None))
```

```
Logistic Regression F1 Score : [0.90554653 0.90362954]
```

In [42]:
```
sns.heatmap(confusion_matrix(test_y, logreg.predict(test_x)), annot=True, 
plt.title("Logistic Regression Confusion Matrix",fontsize=18, color="blue"
```

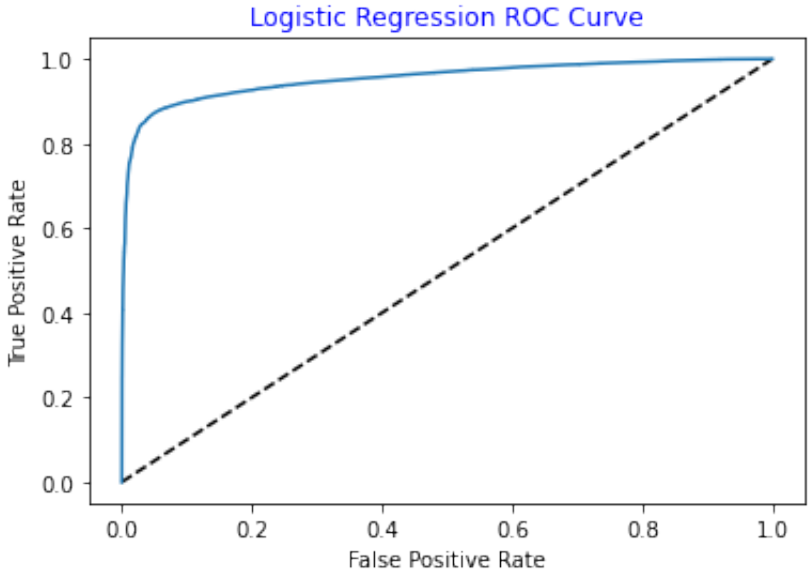


observation

- From the above, we can see that the unbalenced data problem is solved by undersampling as the destribution of TP and TN are balanced and they have high percentage in each column.

7) Logistic Regression Curve

In [43]:
```
from sklearn.metrics import roc_curve
test_y = test_y.replace({2:1,1:0})
# Probabilities
logreg_pred_proba = logreg.predict_proba(test_x)

fpr, tpr, thresholds = roc_curve(test_y, logreg_pred_proba[:,1])
plt.plot([0,1],[0,1],"k--")
plt.plot(fpr, tpr, label = "Logistic Regression")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Logistic Regression ROC Curve", color = 'blue')
plt.show()
```



4. Conclusions

- Even if there are no NaN values, we should check if there are missing values. Some data are unrealistic, such as male is pregnant. These data should be converted.
- In this project, the predict accuracy is about 90%. It means the regression model is good to predict the death situation in this project.
- Do not judge the accuracy of a model arbitrarily, as the data may unbalenced.
- Pneumonia is the most important factor to effect the death rate of patients who infected Covid-19. More attention should be paid on the patients who have pneumonia.