In [2]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn import preprocessing

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

from matplotlib import pyplot as plt

import seaborn as sns
```

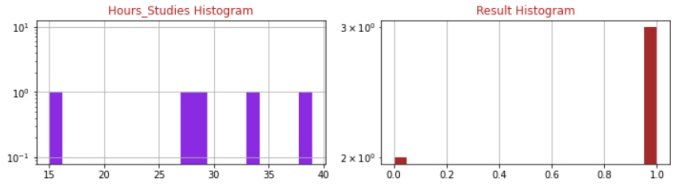
In [3]:
```
df = pd.read_csv('Logistic Regression.csv')
```

In [6]:
```
X = np.array(df['Hours_Studies']).reshape(-1,1)
Y = np.array(df['Result']).reshape(-1,1)
```

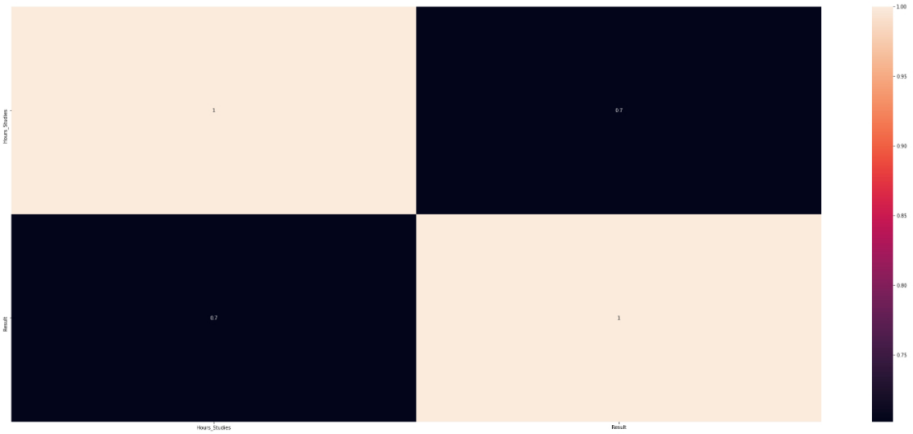
In [15]:
```
X_train, X_test, Y_train, Y_test = train_test_split(
X, Y, test_size=1/3, random_state=5, stratify=y)
```

In [16]:
```
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
```

In [17]:
```
import matplotlib.colors as mcolors
colors = list(mcolors.CSS4_COLORS.keys())[10:]
def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor=colors[i])
        ax.set_title(feature+" Histogram",color=colors[35])
        ax.set_yscale('log')
    fig.tight_layout()
    plt.savefig('Histograms.png')
    plt.show()
draw_histograms(df,df.columns,8,4)
```



In [18]:
```
plt.figure(figsize = (38,16))
sns.heatmap(df.corr(), annot = True)
plt.savefig('heatmap.png')
plt.show()
```



In [19]:
```
model = LogisticRegression()
```

In [20]:
```
model.fit(X_train_scaled, Y_train)
```

```
c:\users\adrij\appdata\local\programs\python\python39\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

Out[20]:
```
LogisticRegression()
```

In [21]:
```
y_pred=model.predict(X_test)
```

In [28]:
```
print('\nThe Coefficient of Logistic Regression Line:',model.coef_[0,0],'\ny-intercept of Logistic Regression Line:',model.ir
```

```
The Coefficient of Logistic Regression Line: 0.28211577235290647
y-intercept of Logistic Regression Line: 0.7071280058546685
```

In [29]:
```
print('\nThe Probability of Pass for the student who studied 33 hrs:',model.predict_proba([[33]])[0,1])
```

```
The Probability of Pass for the student who studied 33 hrs: 0.9999553650391058
```

In [22]:
```python
train_acc = model.score(X_train_scaled, Y_train)
print("The Accuracy for Training Set is {}".format(train_acc*100))
```

```
The Accuracy for Training Set is 66.66666666666666
```

In [23]:
```python
test_acc = accuracy_score(Y_test, y_pred)
print("The Accuracy for Test Set is {}".format(test_acc*100))
```

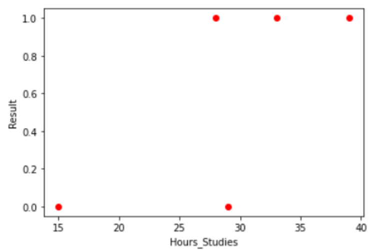
```
The Accuracy for Test Set is 50.0
```

In [25]:
```python
print(classification_report(y_test, y_pred))
```

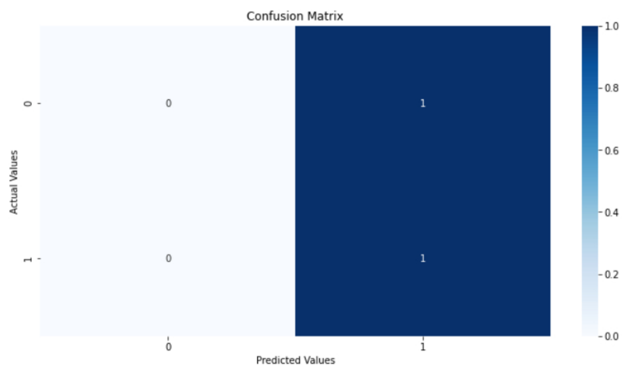
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.50      1.00      0.67         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2
```

```
c:\users\adrij\appdata\local\programs\python\python39\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\users\adrij\appdata\local\programs\python\python39\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\users\adrij\appdata\local\programs\python\python39\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [32]:
```python
plt.xlabel("Hours_Studies")
plt.ylabel("Result")
plt.scatter(df['Hours_Studies'],df['Result'],color='red')
plt.show()
```



In [26]:
```python
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(12,6))
plt.title("Confusion Matrix")
sns.heatmap(cm, annot=True,fmt='d', cmap='Blues')
plt.ylabel("Actual Values")
plt.xlabel("Predicted Values")
plt.savefig('confusion_matrix.png')
```



In [ ]: