In [10]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [12]:
```
dataset = pd.read_csv("Social_Network_Ads.csv")
dataset.head()
```

Out[12]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [14]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [16]:
```
X = dataset.iloc[:, [2, 3]].values   # Age and EstimatedSalary
y = dataset.iloc[:, 4].values        # Purchased
```

In [18]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=0
)
```

In [20]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [22]:
```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

Out[22]:

▾ LogisticRegression

```
LogisticRegression(random_state=0)
```

In [24]:
```
y_pred = classifier.predict(X_test)
```

In [26]:
```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[26]:
```
array([[65,  3],
       [ 8, 24]])
```

In [28]:
```
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]

print("True Positive (TP):", TP)
print("False Positive (FP):", FP)
print("True Negative (TN):", TN)
print("False Negative (FN):", FN)
```

```
True Positive (TP): 24
False Positive (FP): 3
True Negative (TN): 65
False Negative (FN): 8
```

In [30]:
```
accuracy = (TP + TN) / (TP + TN + FP + FN)
accuracy
```

Out[30]:
```
0.89
```

In [32]:
```
error_rate = 1 - accuracy
error_rate
```

Out[32]:
```
0.10999999999999999
```

In [34]:
```
precision = TP / (TP + FP)
precision
```

Out[34]:
```
0.8888888888888888
```

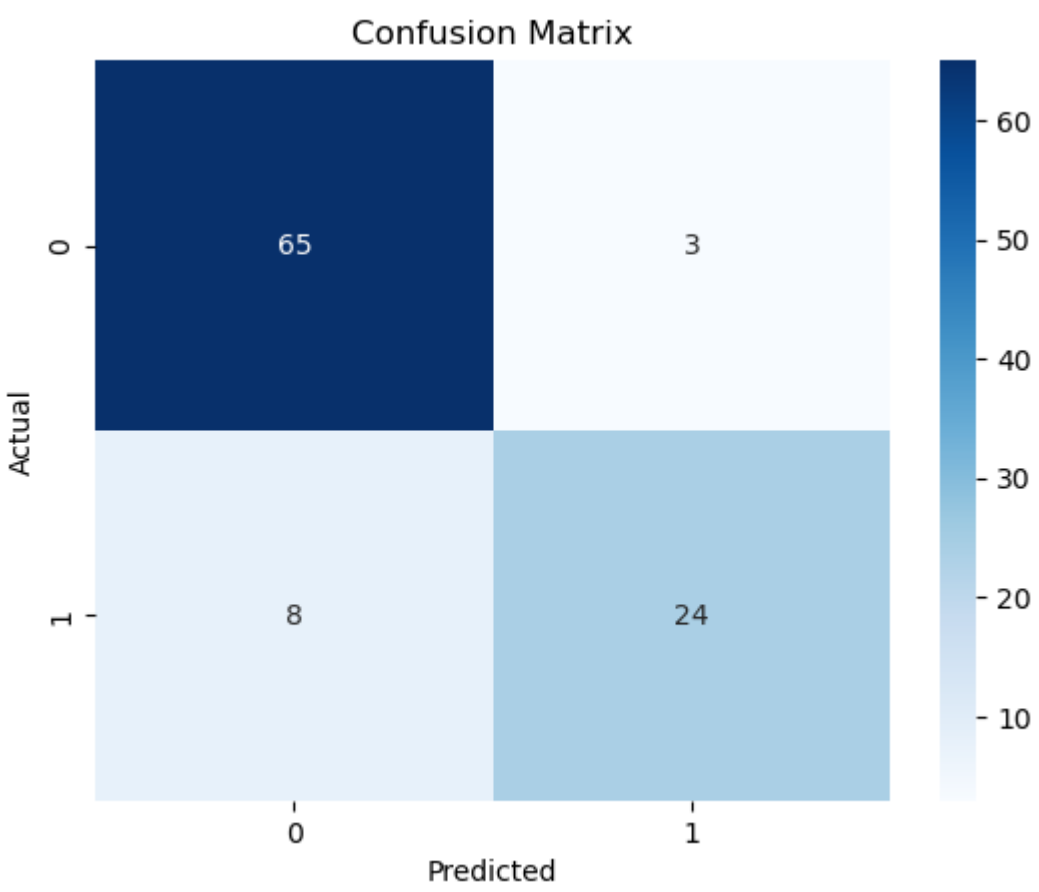
In [36]:
```
recall = TP / (TP + FN)
recall
```

Out[36]:
```
0.75
```

In [38]:
```
print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
print("Precision:", precision)
print("Recall:", recall)
```

```
Accuracy: 0.89
Error Rate: 0.10999999999999999
Precision: 0.8888888888888888
Recall: 0.75
```

In [40]:
```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```


Confusion Matrix
65
3
8
24
0
1
Actual
0
1
Predicted
60
50
40
30
20
10

In [ ]: