In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_m
```

In [2]:
```
data = pd.read_csv("C:\\Users\\Lenovo\\Downloads\\creditcard.csv")
```

In [3]:
```
data
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	C
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	C
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	C
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	C
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-C
...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	C
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	C
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	C
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-C

284807 rows × 31 columns

In [4]:
```
# Data preprocessing
X = data.drop(columns=['Class'])
y = data['Class']
```

In [5]:
```
X
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	C
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	C
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	C
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	C
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-C
...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	C
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	C
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	C
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-C

284807 rows × 30 columns

In [6]:
```
y
```

Out[6]:
```
0         0
1         0
2         0
3         0
```

```
284807 rows × 30 columns
```

In [6]:

```
y
```

Out[6]:

```
0         0
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 284807, dtype: int64
```

In [16]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [17]:

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

In [18]:

```
scaler
```

Out[18]:

```
▾ StandardScaler
StandardScaler()
```

In [19]:

```
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)
```

Out[19]:

```
▾        LogisticRegression
LogisticRegression(random_state=42)
```

In [20]:

```
y_pred = model.predict(X_test)
y_pred
```

Out[20]:

```
array([1, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [21]:

```
print(classification_report(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.86      0.58      0.70        98

    accuracy                           1.00     56962
   macro avg       0.93      0.79      0.85     56962
weighted avg       1.00      1.00      1.00     56962

Confusion Matrix:
[[56855     9]
 [   41    57]]
```

In [22]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

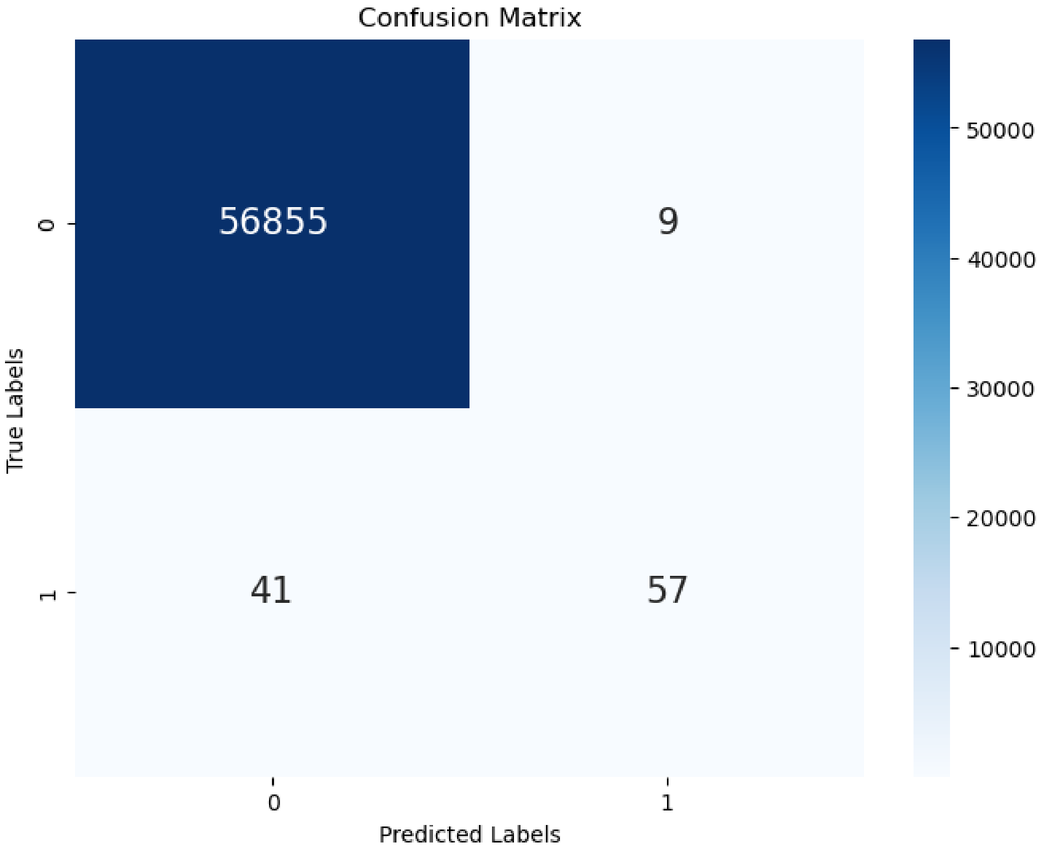
In [23]:

```
# Display a heatmap of the confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

```
 [    41     57]]
```

In [22]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [23]:
```
# Display a heatmap of the confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', annot_kws={'size': 16})
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [24]:
```
from sklearn.metrics import roc_curve, roc_auc_score

# Get predicted probabilities for the positive class (fraud)
y_probs = model.predict_proba(X_test)[:, 1]

# Compute ROC curve and ROC AUC
```

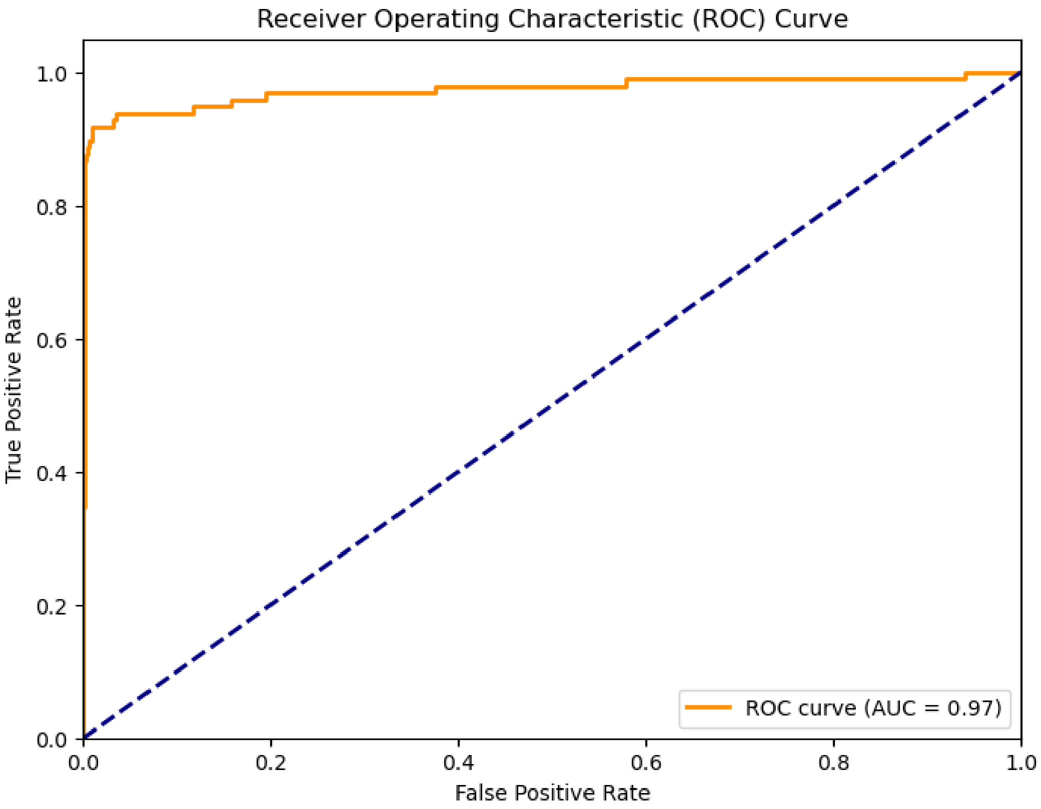
In [24]:

```
from sklearn.metrics import roc_curve, roc_auc_score

# Get predicted probabilities for the positive class (fraud)
y_probs = model.predict_proba(X_test)[:, 1]

# Compute ROC curve and ROC AUC
fpr, tpr, _ = roc_curve(y_test, y_probs)
roc_auc = roc_auc_score(y_test, y_probs)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC curve (AUC = 0.97)