In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:

```
df = pd.read_csv("creditcard.csv")
```

In [3]:

```
df.head()
```

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
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941

5 rows × 31 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [5]:

```
df.describe()
```

Out[5]:

	Time	V1	V2	V3	V
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+0
mean	94813.859575	1.175161e-15	3.384974e-16	-1.379537e-15	2.094852e-1
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+0
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+0
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-0
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-0
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-0
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+0

8 rows × 31 columns

In [6]:

```
df["Class"].value_counts()
```

Out[6]:

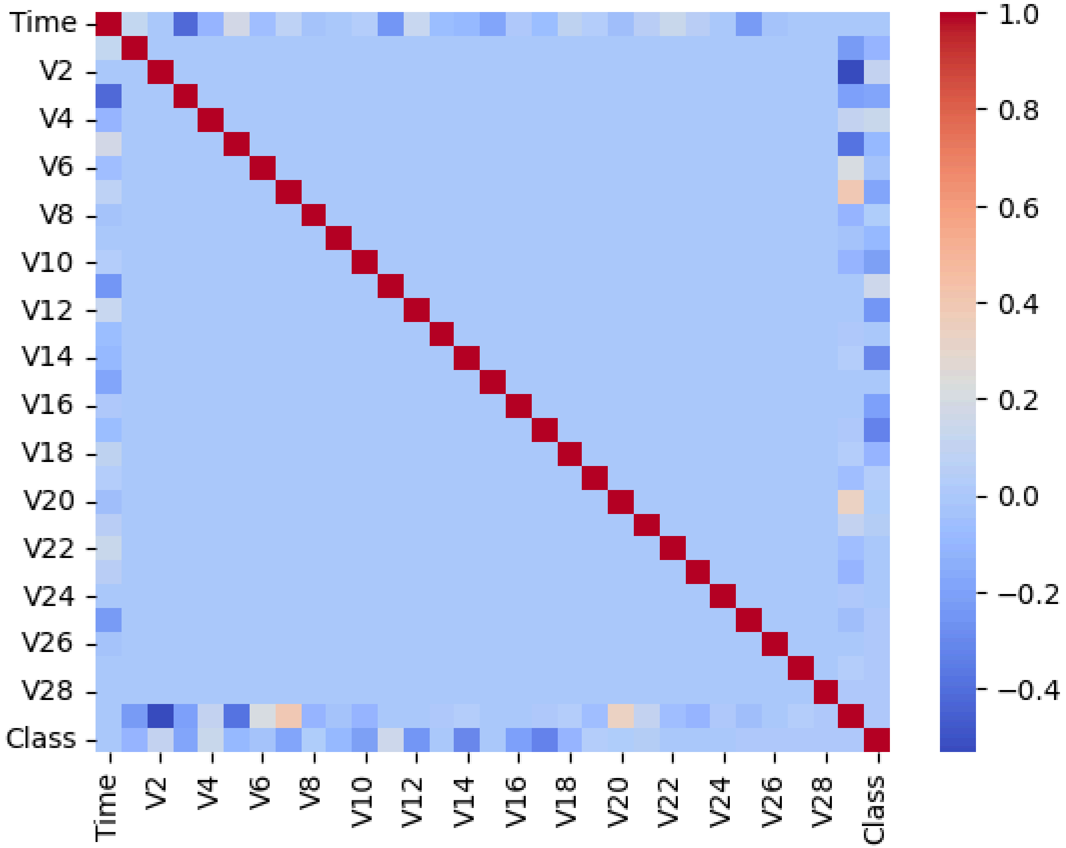
```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [7]:

```
sns.heatmap(df.corr(),cmap='coolwarm')
```

Out[7]:

```
<Axes: >
```



In [8]:

```
df['Class'].value_counts()
```

Out[8]:

```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [9]:

```
normal=df[df.Class==0]
fraud=df[df.Class==1]
print(normal.shape)
print(fraud.shape)
```

```
(284315, 31)
(492, 31)
```

In [10]:

```
df.groupby('Class').mean()
```

Out[10]:

	Time	V1	V2	V3	V4	V5	
Class							
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.0024
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.3977

2 rows × 30 columns

In [11]:

```
normal_sample=normal.sample(n=492)
new_file=pd.concat([normal_sample,fraud],axis=0)
```

In [12]:

```
new_file.head(10)
```

Out[12]:

	Time	V1	V2	V3	V4	V5	V6
30592	35995.0	1.053277	-0.013309	0.282071	1.262380	-0.328526	-0.410608
2806	2364.0	-1.303390	1.451622	0.210419	-0.365147	-0.314256	-0.081985
20625	31164.0	1.534286	-0.938978	-0.739794	-1.907128	-0.060218	0.496395
111760	72358.0	-0.866577	1.577138	1.282551	2.052141	0.711365	1.136185
279506	168914.0	0.036777	0.825349	0.284599	-0.588419	0.401272	-1.056775
13413	23716.0	-1.448540	0.405678	1.549291	1.105241	0.340938	1.706729
252620	155892.0	2.057085	0.156833	-1.794370	1.270376	0.614585	-0.727465
118117	74957.0	-0.884674	0.789085	-0.262496	-2.411150	1.991090	3.151908
237771	149383.0	1.994414	0.198878	-1.527961	1.290176	0.440207	-0.878733
232286	147113.0	1.629657	-1.005061	-0.830974	0.304987	-0.524145	-0.240532

10 rows × 31 columns

In [13]:

```
new_file['Class'].value_counts()
```

Out[13]:

```
Class
0    492
1    492
Name: count, dtype: int64
```

In [14]:

```
new_file.groupby('Class').mean()
```

Out[14]:

	Time	V1	V2	V3	V4	V5	
Class							
0	96173.363821	-0.115545	-0.094846	-0.034031	-0.072742	0.082600	-0.0793
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.3977

2 rows × 30 columns

In [15]:

```
X=new_file.drop(columns='Class',axis=1)
Y=new_file['Class']
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,ra
model=LogisticRegression()
model.fit(X_train,Y_train)
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklear
n\linear_model\_logistic.py:473: ConvergenceWarning: lbfgs failed to converge a
fter 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on
  n_iter_i = _check_optimize_result(
```

Out[15]:

▾ LogisticRegression

Parameters

penalty	'l2'
dual	False
tol	0.0001
C	1.0
fit_intercept	True
intercept_scaling	1
class_weight	None
random_state	None
solver	'lbfgs'
max_iter	100
multi_class	'deprecated'
verbose	0
warm_start	False
n_jobs	None
l1_ratio	None

In [17]:

```
X_train_prediction=model.predict(X_train)
training_data_acuracy=accuracy_score(X_train_prediction,Y_train)*100
print(f"Accuracy of Model: {training_data_acuracy}%")
```

```
Accuracy of Model: 94.79034307496823%
```

In [ ]: