Load the dataset

In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
```

In [5]:

```
df = pd.read_csv('creditcard.csv')
df.head()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278

5 rows × 31 columns

In [6]:

```
# statistical info
df.describe()
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2

8 rows × 31 columns

In [11]:

```
# datatype info
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
```

Preprocessing the dataset

In [13]:

```
# check for null values
df.isnull().sum()
```

Out[13]:

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

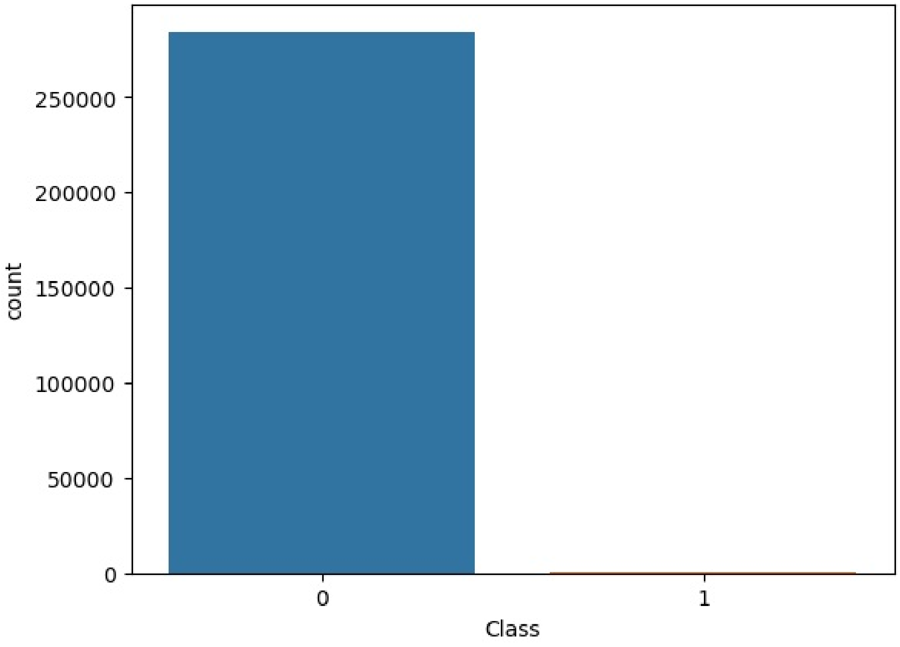
Exploratory Data Analysis

In [15]:

```python
sns.countplot(x='Class', data=df)
```

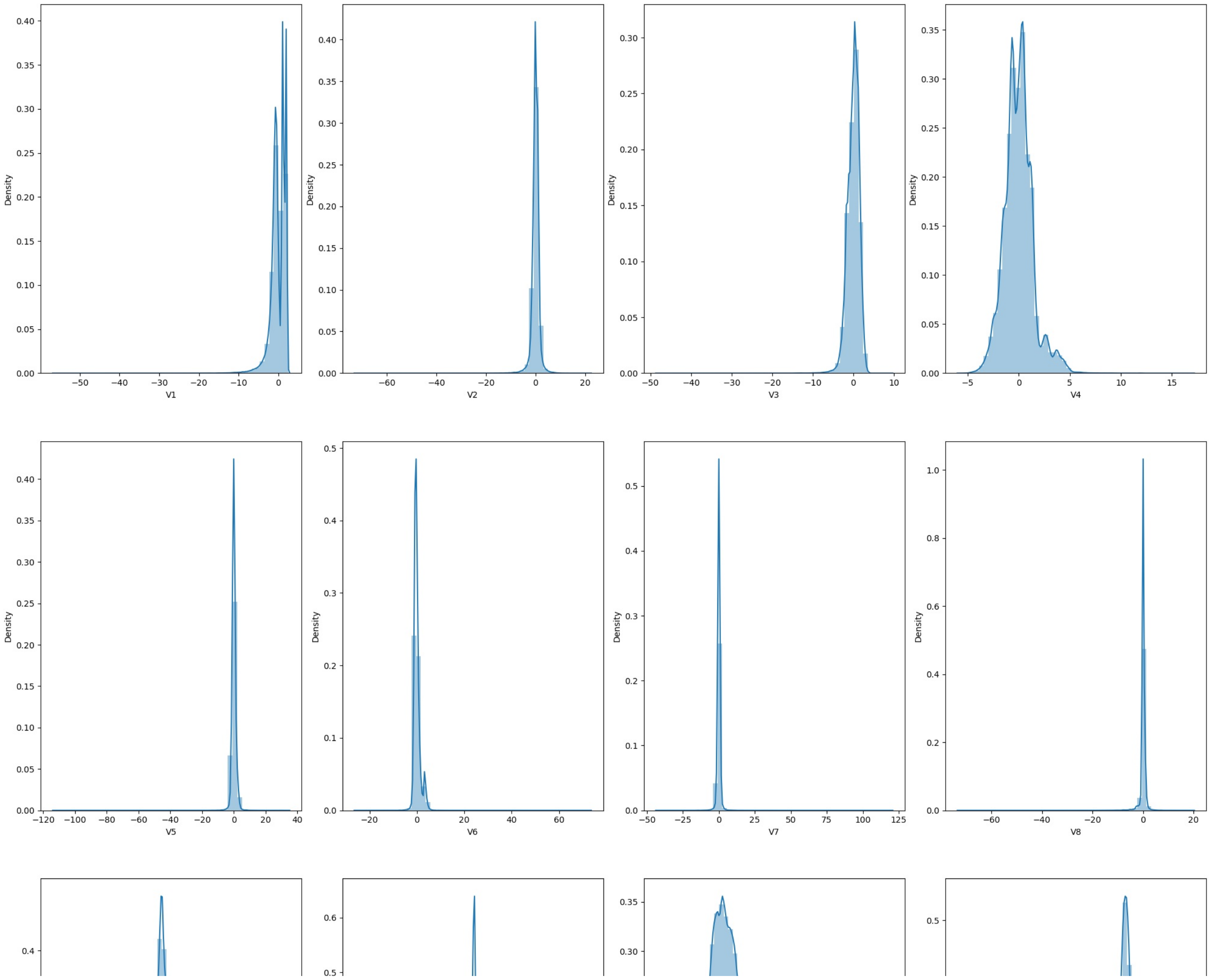
Out[15]:

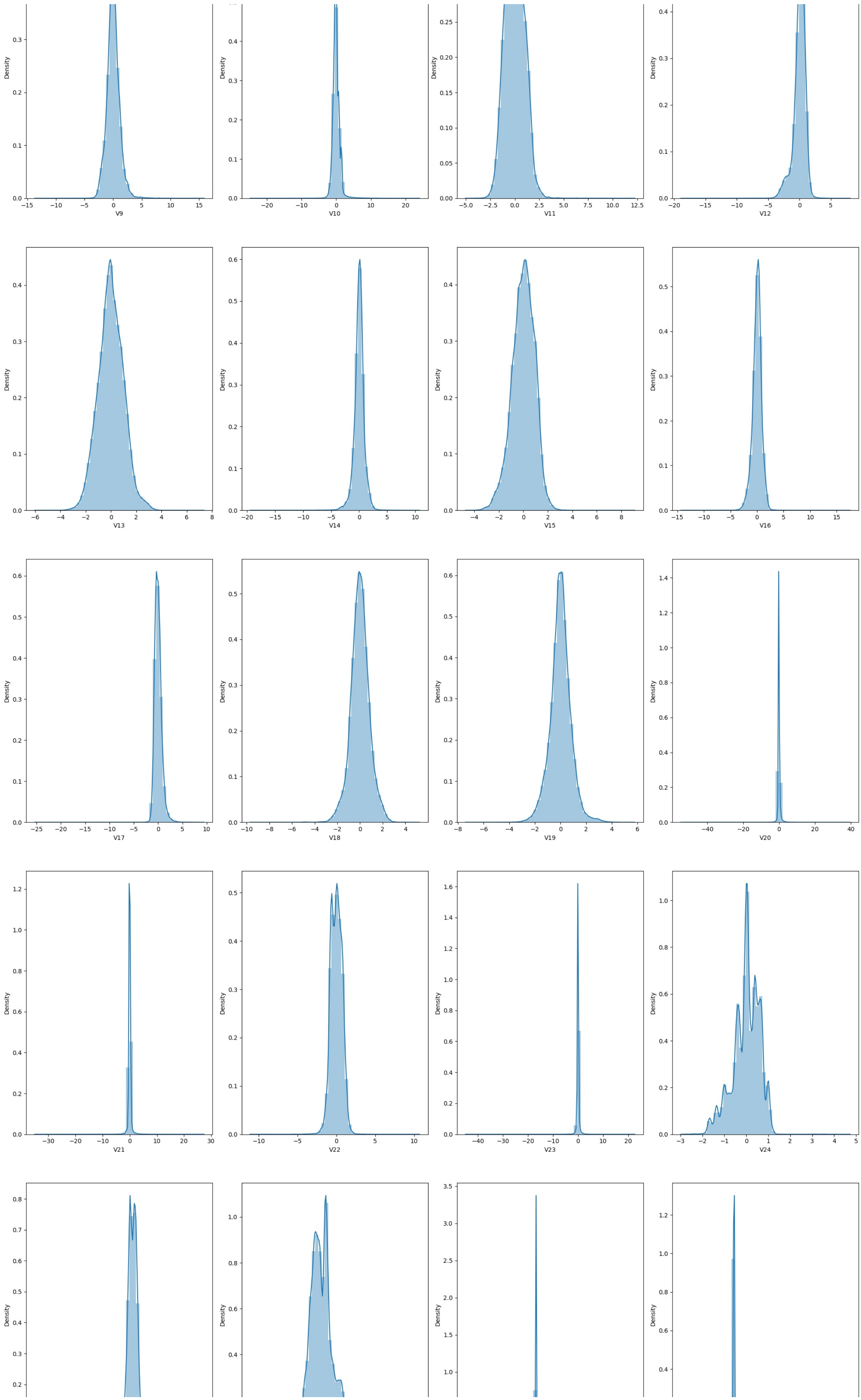
```
<Axes: xlabel='Class', ylabel='count'>
```

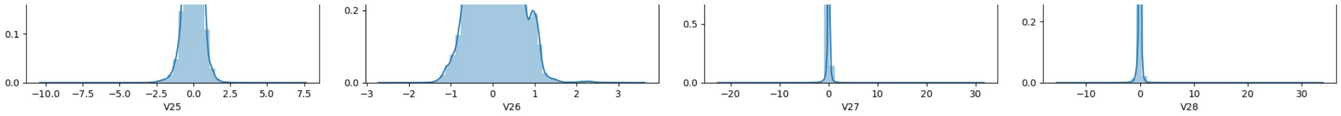


In [17]:

```python
df_temp = df.drop(columns=['Time', 'Amount', 'Class'], axis=1)

# create dist plots
fig, ax = plt.subplots(ncols=4, nrows=7, figsize=(20, 50))
index = 0
ax = ax.flatten()

for col in df_temp.columns:
    sns.distplot(df_temp[col], ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.5, h_pad=5)
```




Density
0.0
0.1
0.2
0.3
−15
−10
−5
0
5
10
15
V9
Density
0.0
0.1
0.2
0.3
0.4
−20
−10
0
10
20
V10
Density
0.00
0.05
0.10
0.15
0.20
0.25
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
V11
Density
0.0
0.1
0.2
0.3
0.4
−20
−15
−10
−5
0
5
V12
Density
0.0
0.1
0.2
0.3
0.4
−6
−4
−2
0
2
4
6
8
V13
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−20
−15
−10
−5
0
5
10
V14
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
6
8
V15
Density
0.0
0.1
0.2
0.3
0.4
0.5
−15
−10
−5
0
5
10
15
V16
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−25
−20
−15
−10
−5
0
5
10
V17
Density
0.0
0.1
0.2
0.3
0.4
0.5
−10
−8
−6
−4
−2
0
2
4
V18
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−8
−6
−4
−2
0
2
4
6
V19
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
−40
−20
0
20
40
V20
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
−30
−20
−10
0
10
20
30
V21
Density
0.0
0.1
0.2
0.3
0.4
0.5
−10
−5
0
5
10
V22
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
−40
−30
−20
−10
0
10
20
V23
Density
0.0
0.2
0.4
0.6
0.8
1.0
−3
−2
−1
0
1
2
3
4
5
V24
Density
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Density
0.4
0.6
0.8
1.0
Density
1.0
1.5
2.0
2.5
3.0
3.5
Density
0.4
0.6
0.8
1.0
1.2

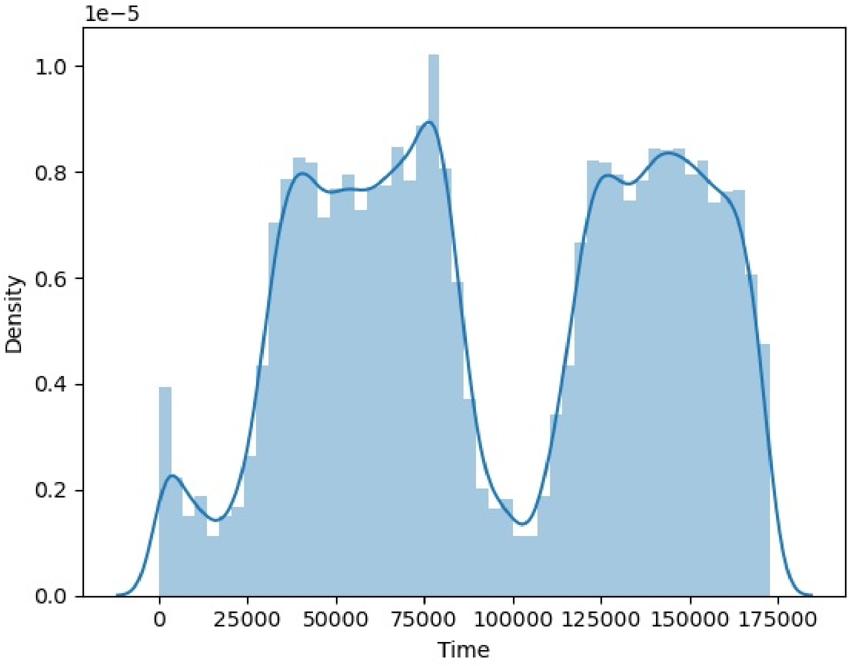


In [18]:

```
sns.distplot(df['Time'])
```

Out[18]:

```
<Axes: xlabel='Time', ylabel='Density'>
```

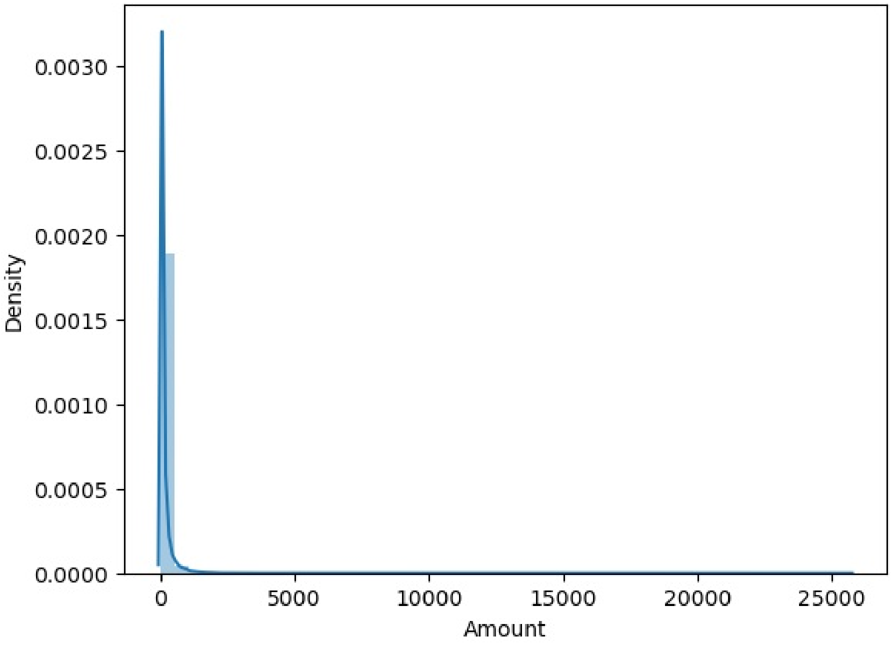


In [19]:

```
sns.distplot(df['Amount'])
```

Out[19]:

```
<Axes: xlabel='Amount', ylabel='Density'>
```



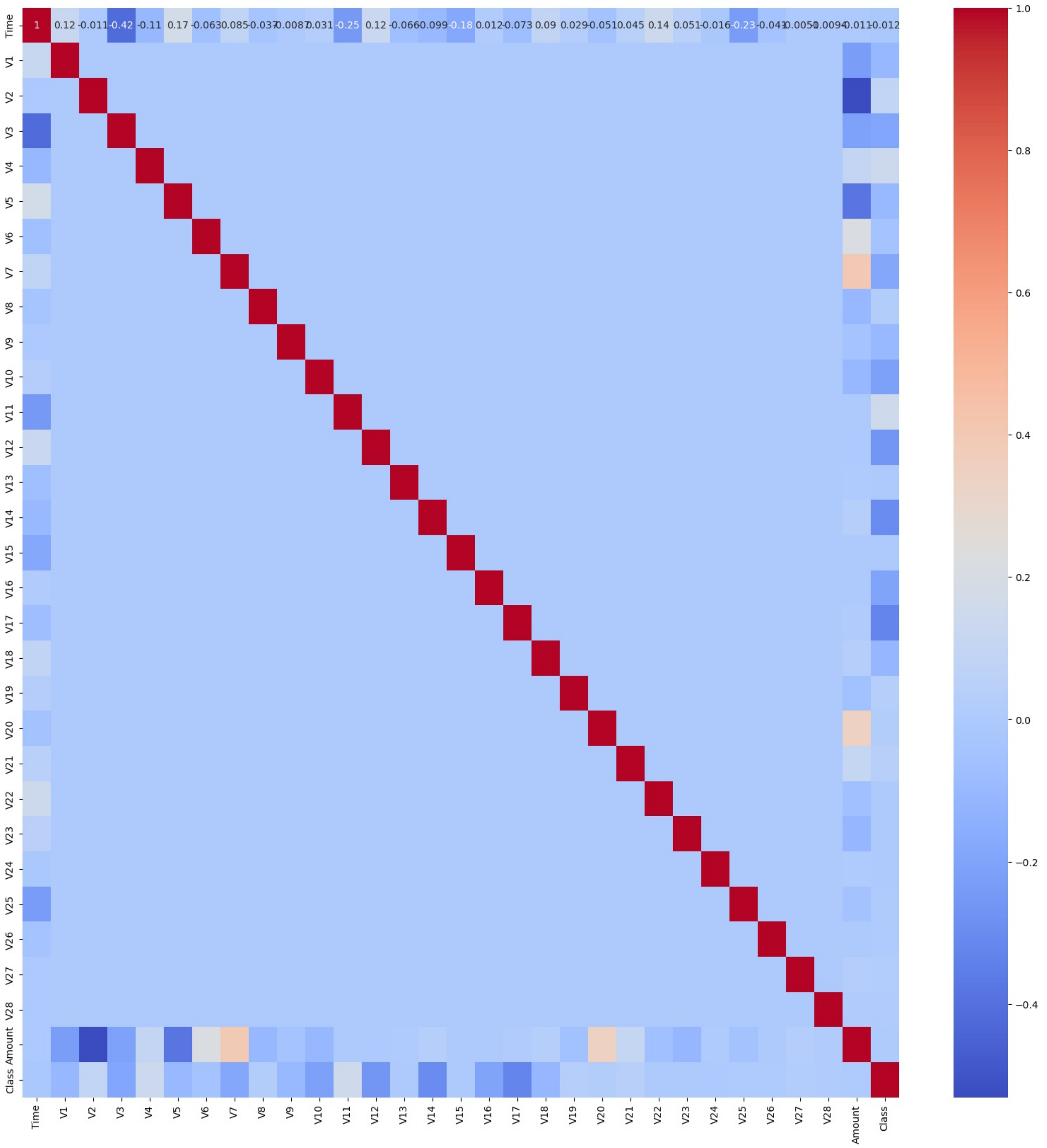
Coorelation Matrix

In [20]:

```
corr = df.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[20]:

```
<Axes: >
```



Input Split

In [25]:

```
X = df.drop(columns=['Class'], axis=1)
y = df['Class']
```

Standard Scaling

In [27]:

```
sc = StandardScaler()
x_scaler = sc.fit_transform(X)
```

In [29]:

```
x_scaler[-1]
```

```
Out[29]: array([ 1.64205773, -0.27233093, -0.11489898,  0.46386564, -0.35757   ,
        -0.00908946, -0.48760183,  1.27476937, -0.3471764 ,  0.44253246,
        -0.84072963, -1.01934641, -0.0315383 , -0.18898634, -0.08795849,
         0.04515766, -0.34535763, -0.77752147,  0.1997554 , -0.31462479,
         0.49673933,  0.35541083,  0.8861488 ,  0.6033653 ,  0.01452561,
        -0.90863123, -1.69685342, -0.00598394,  0.04134999,  0.51435531])
```

Model Training

```
In [31]: x_train, x_test, y_train, y_test = train_test_split(x_scaler, y, test_size=0.25, random_state=42, stratify=y)
```

```
In [33]: model = LogisticRegression()
         # training
         model.fit(x_train, y_train)
         # testing
         y_pred = model.predict(x_test)
         print(classification_report(y_test, y_pred))
         print("F1 Score:",f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.84      0.62      0.71       123

    accuracy                           1.00     71202
   macro avg       0.92      0.81      0.85     71202
weighted avg       1.00      1.00      1.00     71202

F1 Score: 0.7102803738317757
```

```
In [35]: model = RandomForestClassifier()
         # training
         model.fit(x_train, y_train)
         # testing
         y_pred = model.predict(x_test)
         print(classification_report(y_test, y_pred))
         print("F1 Score:",f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.95      0.79      0.86       123

    accuracy                           1.00     71202
   macro avg       0.98      0.89      0.93     71202
weighted avg       1.00      1.00      1.00     71202

F1 Score: 0.8622222222222222
```

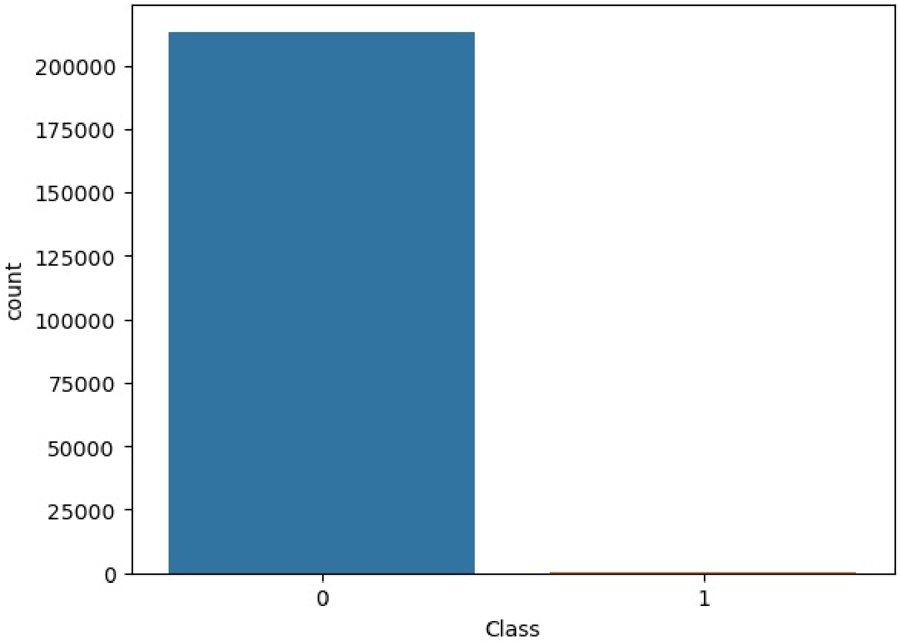
```
In [39]: model = XGBClassifier(n_jobs=-1)
         # training
         model.fit(x_train, y_train)
         # testing
         y_pred = model.predict(x_test)
         print(classification_report(y_test, y_pred))
         print("F1 Score:",f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.95      0.79      0.86       123

    accuracy                           1.00     71202
   macro avg       0.98      0.89      0.93     71202
weighted avg       1.00      1.00      1.00     71202

F1 Score: 0.8622222222222222
```

class Imbalancement

```
In [41]: sns.countplot(x=y_train,data=df)
```

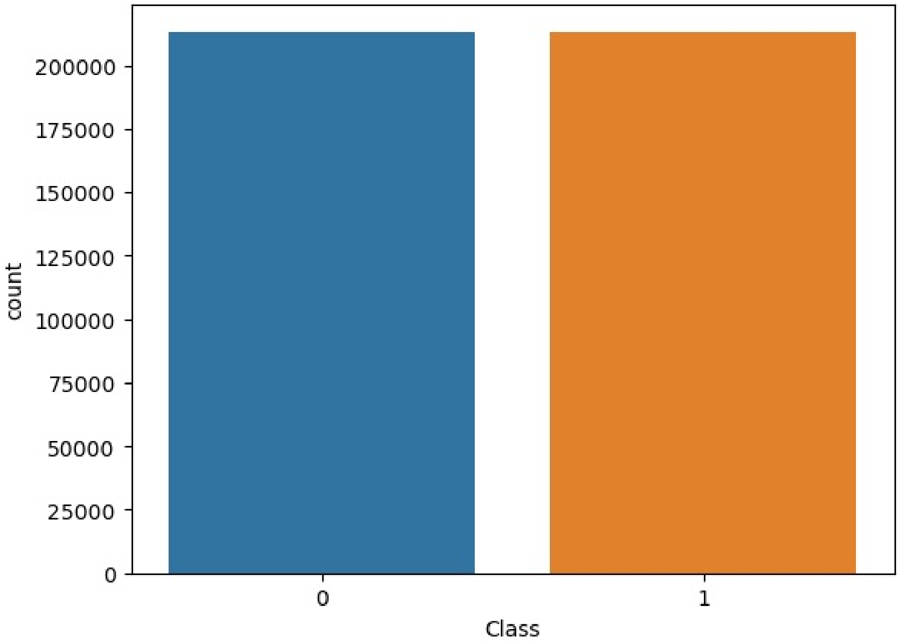
```
Out[41]: <Axes: xlabel='Class', ylabel='count'>
```



```
In [43]: over_sample = SMOTE()
         x_smote, y_smote = over_sample.fit_resample(x_train, y_train)
```

```
In [45]: sns.countplot(x=y_smote,data=df)
```

```
Out[45]: <Axes: xlabel='Class', ylabel='count'>
```



```
In [47]: model = LogisticRegression()
         # training
         model.fit(x_smote, y_smote)
         # testing
         y_pred = model.predict(x_test)
         print(classification_report(y_test, y_pred))
         print("F1 Score:",f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     71079
           1       0.06      0.89      0.11       123

    accuracy                           0.98     71202
   macro avg       0.53      0.93      0.55     71202
weighted avg       1.00      0.98      0.99     71202

F1 Score: 0.11139499233520694
```

In [49]:

```
model = RandomForestClassifier(n_jobs=-1)
# training
model.fit(x_smote, y_smote)
# testing
y_pred = model.predict(x_test)
print(classification_report(y_test, y_pred))
print("F1 Score:",f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.89      0.80      0.84       123

    accuracy                           1.00     71202
   macro avg       0.95      0.90      0.92     71202
weighted avg       1.00      1.00      1.00     71202

F1 Score: 0.8412017167381974
```

In [51]:

```
model = XGBClassifier(n_jobs=-1)
# training
model.fit(x_smote, y_smote)
# testing
y_pred = model.predict(x_test)
print(classification_report(y_test, y_pred))
print("F1 Score:",f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.75      0.84      0.79       123

    accuracy                           1.00     71202
   macro avg       0.88      0.92      0.90     71202
weighted avg       1.00      1.00      1.00     71202

F1 Score: 0.7923076923076923
```

In [53]:

```
# created by Piyush Verma as Data Scientist at Growintern
# For contact pverma20269968@gmail.com
```

In [ ]: