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

In [3]:
```
df = pd.read_csv('Desktop\creditcard.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

In [5]:
```
df = df.drop(['Time', 'Class'], axis = 1)
df.shape
```

Out[5]:
```
(284807, 29)
```

In [6]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df.iloc[:, :-1] = scaler.fit_transform(df.iloc[:, :-1])
```

In [7]:
```
df.head()
```

Out[7]:

	V1	V2	V3	V4	V5	V6	V7	V8	
0	-0.694242	-0.044075	1.672773	0.973366	-0.245117	0.347068	0.193679	0.082637	0.33
1	0.608496	0.161176	0.109797	0.316523	0.043483	-0.061820	-0.063700	0.071253	-0.23
2	-0.693500	-0.811578	1.169468	0.268231	-0.364572	1.351454	0.639776	0.207373	-1.37
3	-0.493325	-0.112169	1.182516	-0.609727	-0.007469	0.936150	0.192071	0.316018	-1.26
4	-0.591330	0.531541	1.021412	0.284655	-0.295015	0.071999	0.479302	-0.226510	0.74

5 rows × 29 columns

In [8]:
```
from sklearn.model_selection import train_test_split

x_train, x_test = train_test_split(df, test_size=0.2)

print("x_train shape : ", x_train.shape)
print("x_test shape : ", x_test.shape)
```

```
x_train shape :  (227845, 29)
x_test shape :  (56962, 29)
```

In [9]:
```
from keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import layers, models

encoder = tf.keras.models.Sequential([
    layers.Input(shape=(x_train.shape[1],)),
    layers.Dense(20, activation='relu'),
    layers.Dense(20, activation='relu'),
    layers.Dense(10, activation='relu')
])

decoder = tf.keras.models.Sequential([
    layers.Input(shape=(10,)),
    layers.Dense(20, activation='relu'),
    layers.Dense(20, activation='relu'),
    layers.Dense(x_train.shape[1], activation='linear')
])

model = tf.keras.models.Sequential([
    encoder,
    decoder,
])
```

In [10]:
```
model.compile(optimizer='adam', loss ='mean_squared_error')
```

In [11]:
```
history = model.fit(
    x_train,
    x_train,
    validation_data=(x_test,x_test),
    epochs=10,
    batch_size = 500,
    shuffle=True
)
```

```
Epoch 1/10
456/456 [==============================] - 4s 6ms/step - loss: 223.8373 -
val_loss: 1.0004
Epoch 2/10
456/456 [==============================] - 2s 5ms/step - loss: 0.8491 - v
al_loss: 0.8041
Epoch 3/10
456/456 [==============================] - 2s 5ms/step - loss: 0.7655 - v
al_loss: 0.7357
Epoch 4/10
456/456 [==============================] - 2s 5ms/step - loss: 0.6828 - v
al_loss: 0.7777
Epoch 5/10
456/456 [==============================] - 2s 4ms/step - loss: 0.7826 - v
al_loss: 0.6901
Epoch 6/10
456/456 [==============================] - 2s 5ms/step - loss: 0.6614 - v
al_loss: 0.6123
Epoch 7/10
456/456 [==============================] - 2s 4ms/step - loss: 0.7350 - v
al_loss: 0.6275
Epoch 8/10
456/456 [==============================] - 1s 3ms/step - loss: 0.6138 - v
al_loss: 0.6003
Epoch 9/10
456/456 [==============================] - 1s 3ms/step - loss: 1.0103 - v
al_loss: 0.8047
Epoch 10/10
456/456 [==============================] - 2s 5ms/step - loss: 0.6182 - v
al_loss: 0.5644
```

In [ ]:
```
```

In [13]:
```
predictions = model.predict(x_test)
```

```
1781/1781 [==============================] - 3s 2ms/step
```

In [14]:
```
mse = np.mean(np.power(x_test - predictions, 2), axis=1)
mse
```

Out[14]:
```
49994     0.253195
194137    0.421602
80151     0.387397
163520    0.305153
180985    0.347469
             ...
221778    0.363021
3480      0.230793
207647    0.843475
138889    0.244909
72395     0.161340
Length: 56962, dtype: float64
```

In [15]:
```
threshold = np.percentile(mse, 95)
threshold
```

Out[15]:
```
1.1053713304779724
```

In [16]:
```
anomalies = mse > threshold
```

In [17]:
```
num_anomalies = np.sum(anomalies)
print(f"Number of Anomalies: {num_anomalies}")
```

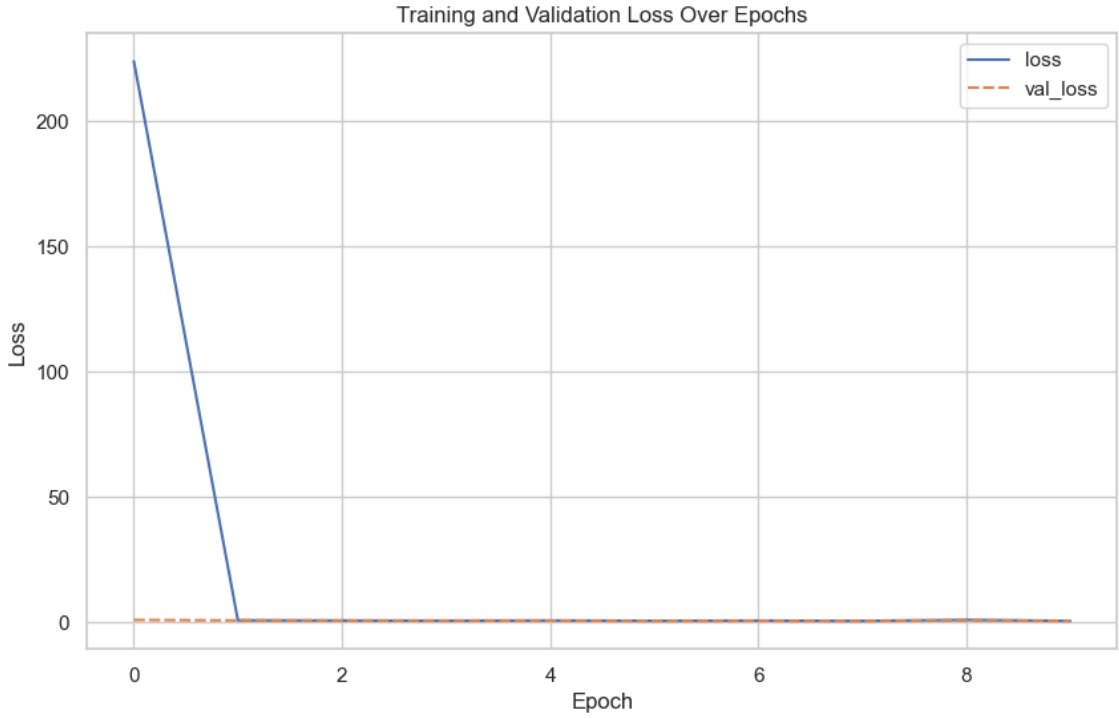
```
Number of Anomalies: 2849
```

In [25]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
loss_plot = sns.lineplot(data=history.history)

# Set plot labels and title
loss_plot.set(xlabel='Epoch', ylabel='Loss', title='Training and Validation

plt.show()
```



In [ ]:
```
```