IESB

Pós Graduação em Inteligência Artificial

Disciplina: Estatística e Análise de dados

Docente: Mateus Mendelson

Discente: Henrique Brandão

Credit card

In [213]:
```
import scipy
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt

from seaborn_qqplot import pplot
from sklearn.utils import shuffle
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

%config IPCompleter.use_jedi = False
%matplotlib notebook
```

In [2]:
```
df = pd.read_csv('creditcard.csv')
```

In [3]:
```
df.head()
```

	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	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-(

5 rows × 31 columns

In [4]:
```
def _isnull(df):
    for c in df.columns:
        if df[pd.isnull(df[c])].shape[0] != 0:
            print(f'# {c}: {cnt} rows')

_isnull(df)
```

In [5]:
```
df['Class'].value_counts()
```

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

In [124]:
```
print('Índice de fraudes: {:.2f}%'.format((492/284315)*100))
```

```
Índice de fraudes: 0.17%
```

In [6]:
```
df['Amount'].describe()
```

```
count    284807.000000
mean         88.349619
std         250.120109
min           0.000000
25%           5.600000
50%          22.000000
75%          77.165000
max       25691.160000
Name: Amount, dtype: float64
```

In [144]:
```
legit = sum(df['Amount'][df['Class'] == 0])
frd = sum(df['Amount'][df['Class'] == 1])

legit, frd, legit - frd, legit/frd
```

```
(25102462.039983638, 60127.96999999997, 25042334.06998364, 417.4839436618873)
```

In [143]:
```
print('Percentual do montante fraudado: {:.2f}%'.format((frd/legit)
```

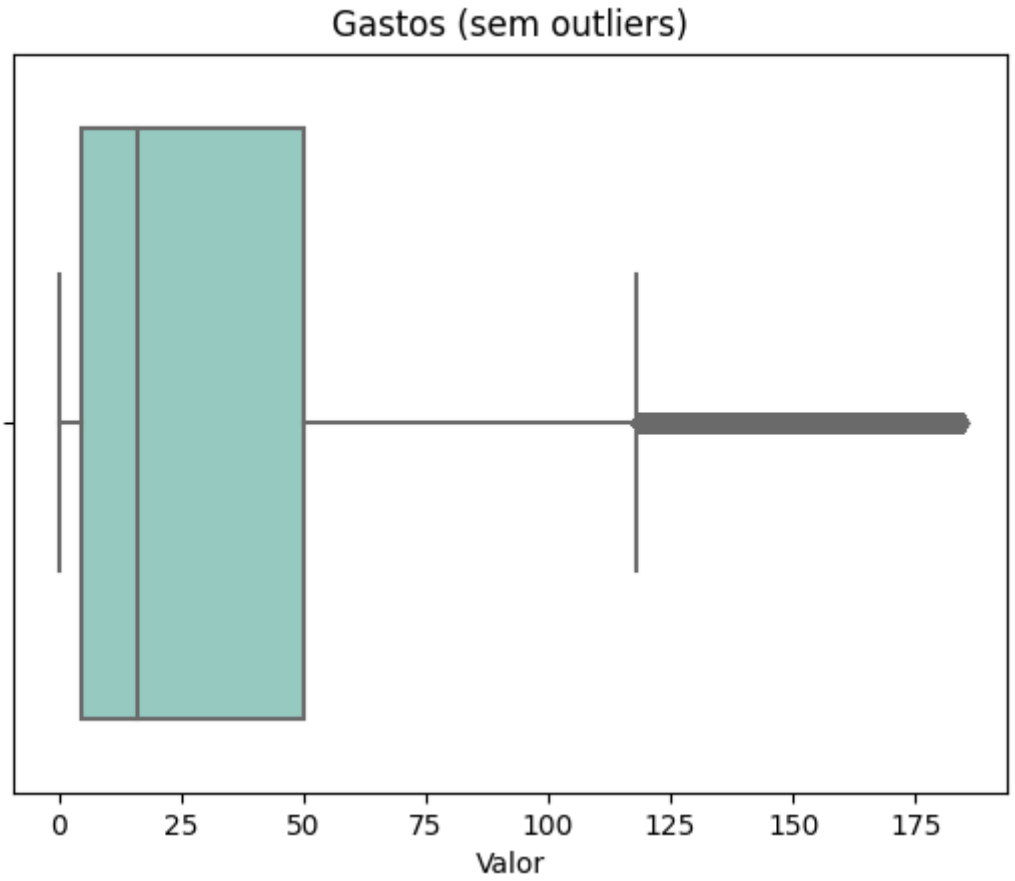
```
Percentual do montante fraudado: 0.24%
```

In [307]:
```
q1 = 5.6
q3 = 77.165
iqr = q3 - q1

inf = q1 - 1.5*iqr
sup = q3 + 1.5*iqr
```

```
        def not_outlier(x):
            if (x >= inf) and (x <= sup):
                return True
            else:
                return False
```

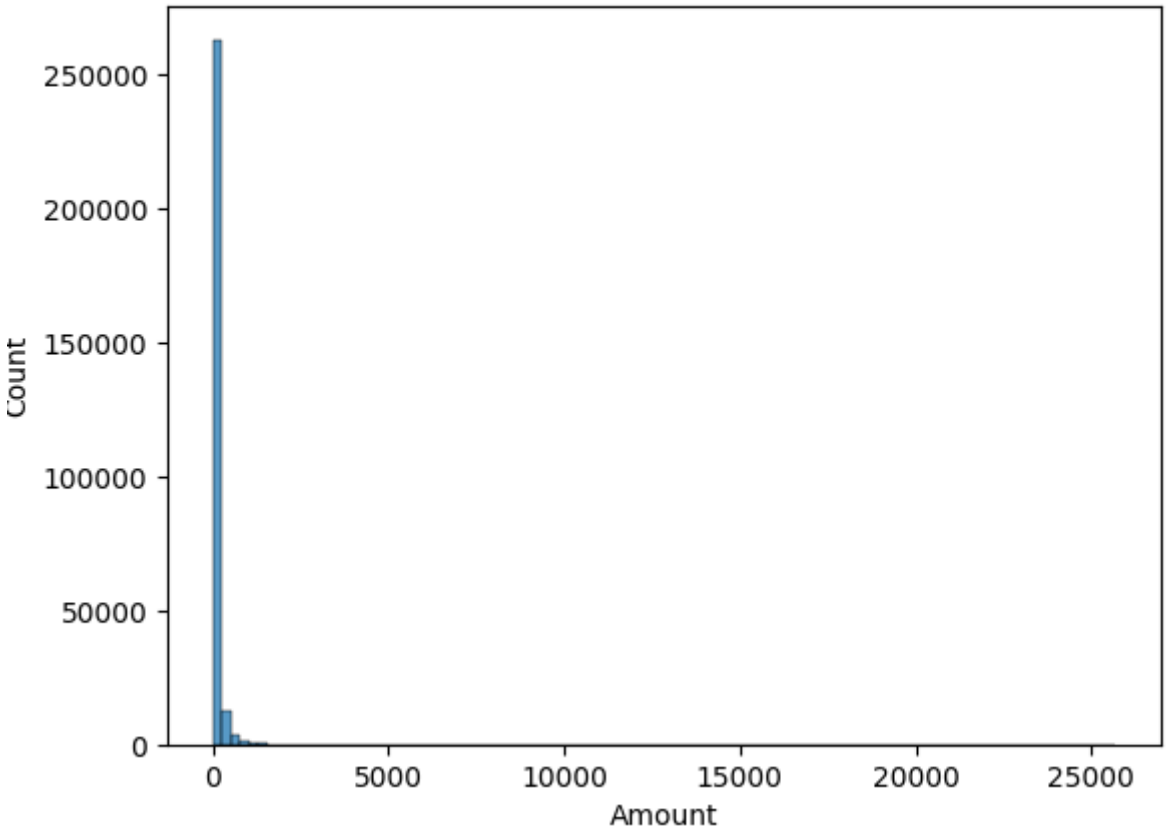
In [311]:
```
amount_sem_out = list(filter(not_outlier, df['Amount']))
```

In [316]:
```
sns.boxplot(x=amount_sem_out, palette="Set3")
plt.title('Gastos (sem outliers)')
plt.xlabel('Valor')
plt.show()
```



1) Visualização do histograma de gastos geral do dataset (at
adequada do parâmetro **bins**)

In [297]:
```
sns.histplot(data=df['Amount'], bins=100)
plt.show()
```



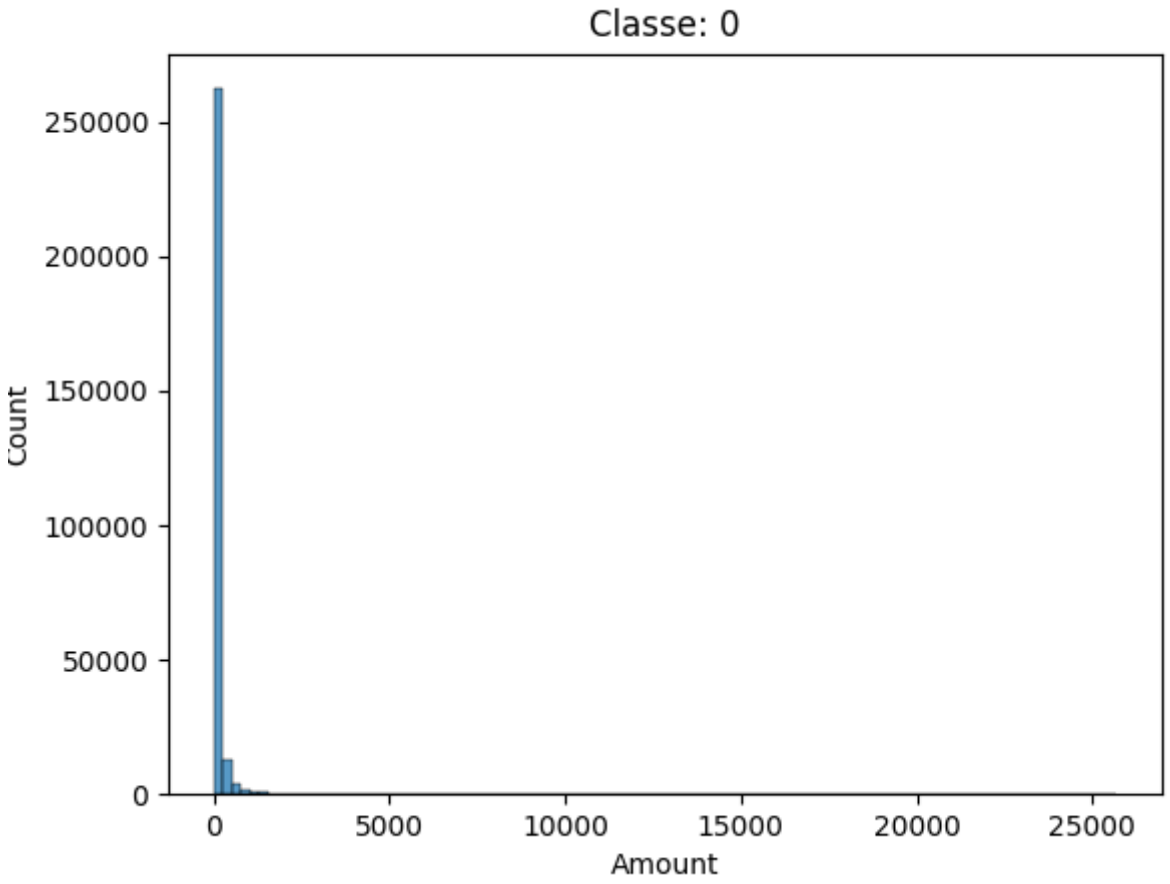
2) Visualização dos histogramas de gastos das transações fra transações não fraudulentas em separado (atente-se para ur parâmetro **bins**)

In [14]:
```
# Legit
df['Amount'][df['Class'] == 0].describe()
```

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

In [133]:
```
sns.histplot(data=df['Amount'][df['Class'] == 0], bins=100)
plt.title('Classe: 0')
```

```
plt.show()
```



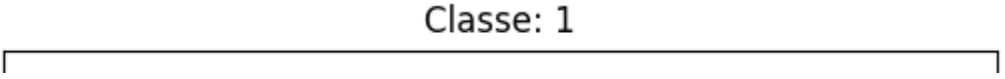
In [134]:

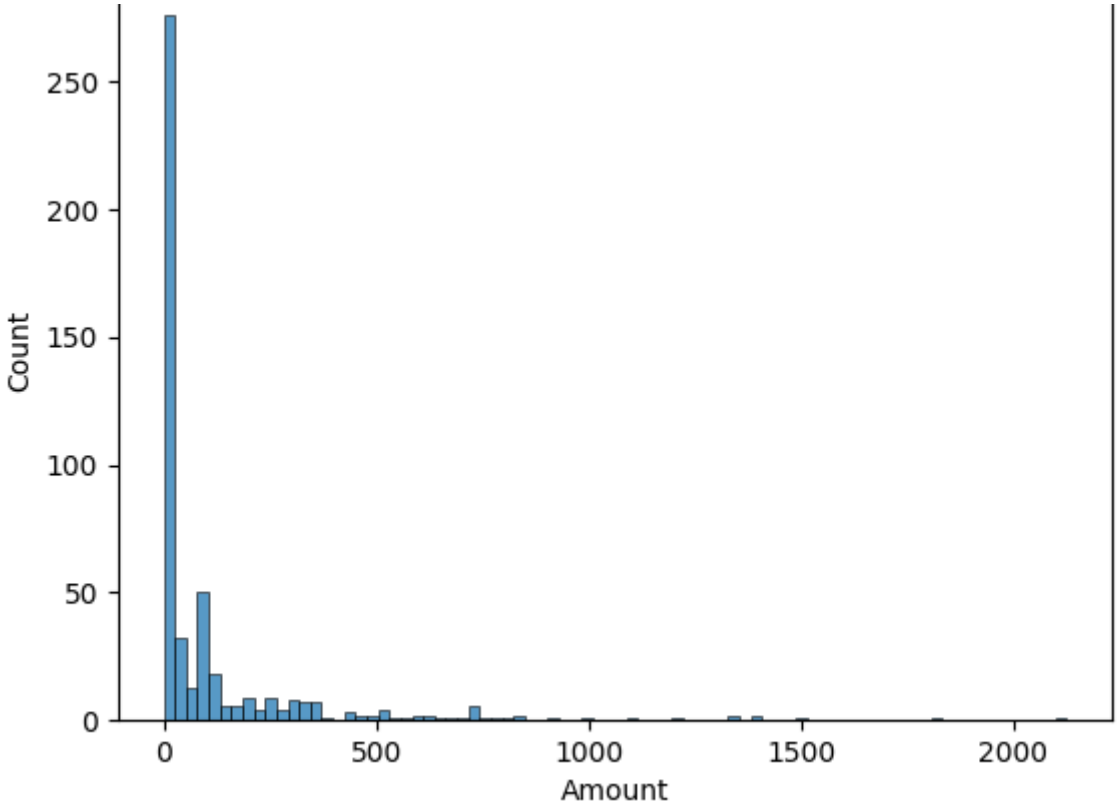
```
# Fraud
df['Amount'][df['Class'] == 1].describe()
```

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

In [135]:

```
sns.histplot(data=df['Amount'][df['Class'] == 1], bins=80)
plt.title('Classe: 1')
plt.show()
```





3) Para cada histograma gerado, indique, utilizando QQ-Plot, adequa a ele e caracterize a distribuição (média e desvio pad

Vamos normalizar as distribuições, através da seguinte relação: $z =$

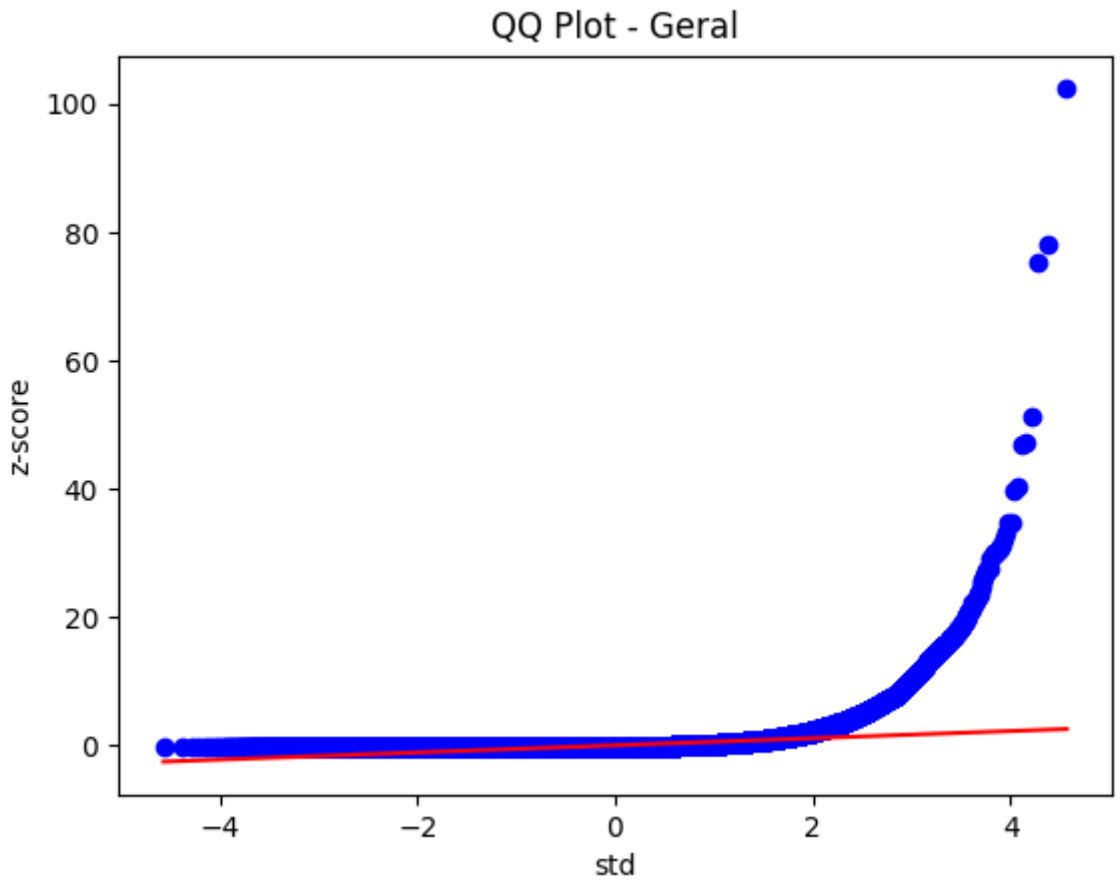
In [30]:
```
def normalizar_df(df, col: str):
    pop = df[col]
    media = df[col].mean()
    std = df[col].std()
    return (pop - media)/std
```

- Geral

In [199]:
```
df_geral = df['Amount']
geral_media = df_geral.mean()
geral_std = df_geral.std()

z_geral = (df_geral - geral_media)/geral_std

print('Média: {:.2f}, desvio padrão: {:.2f}'.format(geral_media, gera

scipy.stats.probplot(z_geral, plot=plt, )
plt.title('QQ Plot - Geral')
```

```
plt.ylabel('z-score')
plt.xlabel('std')
plt.show()
```

```
Média: 88.35, desvio padrão: 250.12
```



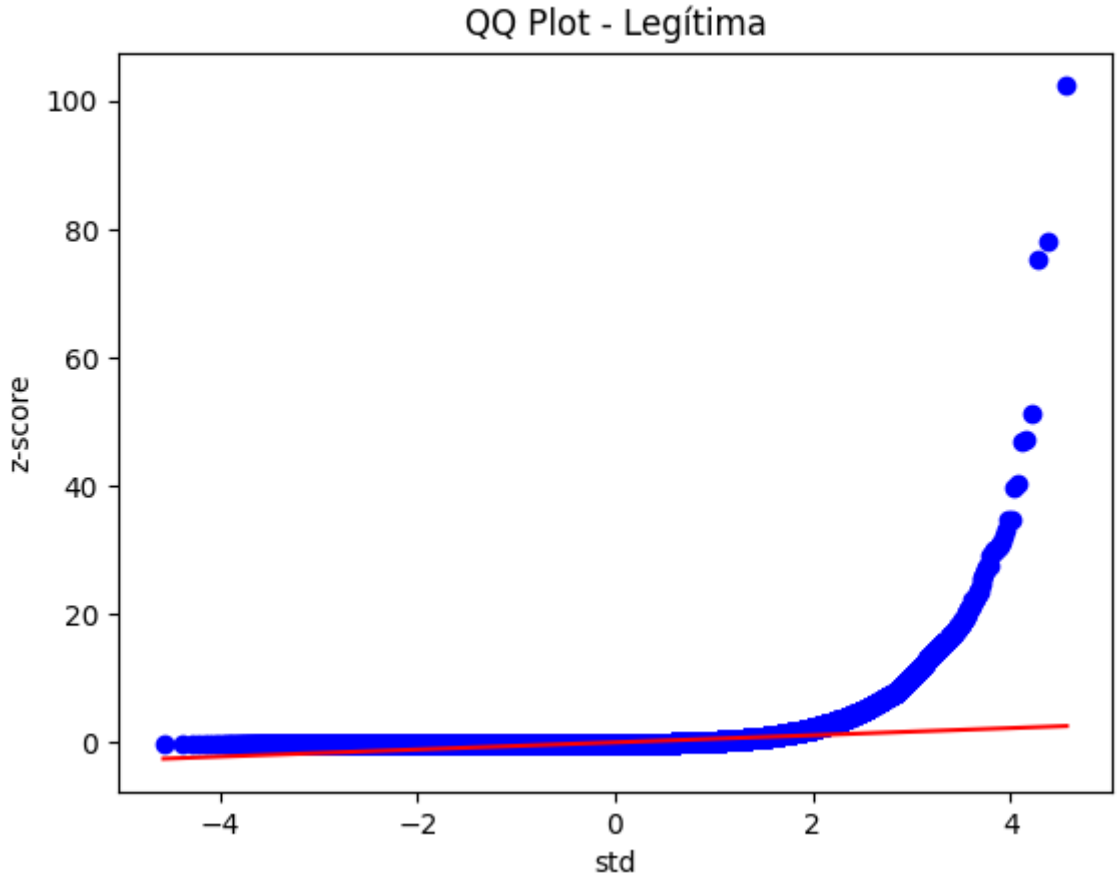
- Legítima

In [200]:

```
df_legit = df['Amount'][df['Class'] == 0]
legit_media = df_legit.mean()
legit_std = df_legit.std()

z_legit = (df_legit - legit_media)/legit_std

print('Média: {:.2f}, desvio padrão: {:.2f}'.format(legit_media, leg

scipy.stats.probplot(z_legit, dist="norm", plot=plt)
plt.title('QQ Plot - Legítima')
plt.ylabel('z-score')
plt.xlabel('std')
```

```
plt.show()
```

```
Média: 88.29, desvio padrão: 250.11
```

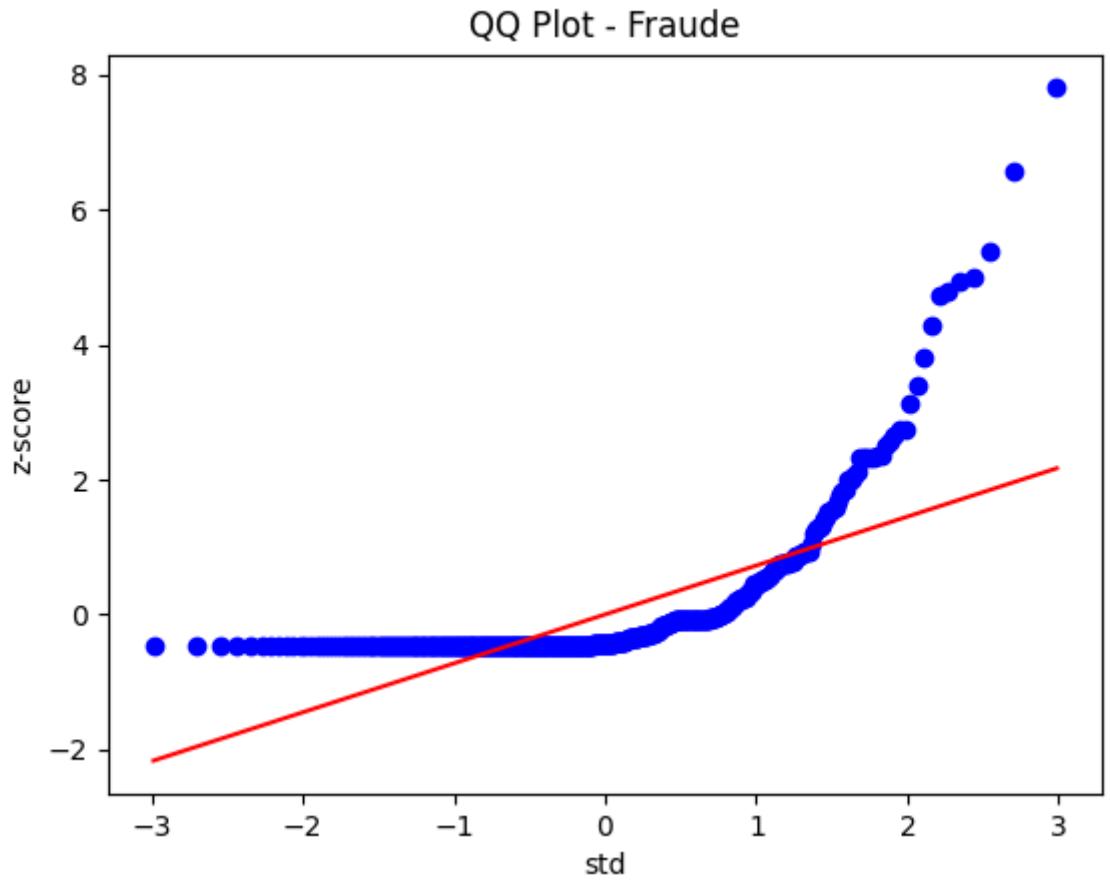


- Fraude

In [197]:

```
df_fraud = df['Amount'][df['Class'] == 1]
fraud_media = df_fraud.mean()
fraud_std = df_fraud.std()

z_fraud = (df_fraud - fraud_media)/fraud_std

print('Média: {:.2f}, desvio padrão: {:.2f}'.format(fraud_media, frau

scipy.stats.probplot(z_fraud, dist="norm", plot=plt)
plt.title('QQ Plot - Fraude')
plt.ylabel('z-score')
plt.xlabel('std')
plt.show()
```

```
Média: 122.21, desvio padrão: 256.68
```



4) Realize um teste A/B da seguinte forma: separe, aleatoriar
grupo A deve conter 50% das transações não fraudulentas e
fraudulentas. O grupo B, por sua vez, deve conter os dados r
dataset tivesse 100 transações (sendo 80 não fraudulentas e

> A teria 40 transações não fraudulentas e 10 transações frauc
> quantidades se aplicariam ao grupo B). O grupo A deve ser o
> grupo B será o grupo que receberá a nova técnica de identifi
> fraudulentas. Essa técnica consiste em: caso a transação pos
> igual a 122.20, ela deve ser classificada como fraudulenta. A
> nova técnica sobre a porcentagem de transações fraudulenta
> porcentagem das transações que realmente eram fraudulent
> classificadas como fraudulentas? A técnica aplicada no grupo
> grupo A?

In [21]:
```
sdf = shuffle(df[['Amount', 'Class']])
sdf.head()
```

	Amount	Class
175842	1.51	0
182727	25.00	0
172416	42.00	0
16581	3.80	0
120017	0.01	0

In [22]:
```
sdf_0 = sdf[sdf['Class'] == 0]
sdf_1 = sdf[sdf['Class'] == 1]

sdf_0.shape[0], sdf_1.shape[0],
```

```
(284315, 492)
```

In [23]:
```
metade_0 = round(sdf_0.shape[0]/2)
metade_1 = round(sdf_1.shape[0]/2)

metade_0, metade_1
```

```
(142158, 246)
```

In [24]:
```
A = pd.concat([sdf_0[:metade_0], sdf_1[:metade_1]])
A.shape
```

```
(142404, 2)
```

In [153]:
```
B = pd.concat([sdf_0[metade_0:], sdf_1[metade_1:]])
B.shape
```

```
(142403, 2)
```

In [154]:
```
A['Class'].value_counts()
```

```
0    142158
1       246
Name: Class, dtype: int64
```

In [155]:
```
perc_fraudes_em_A = (A['Class'].value_counts()[1]/A.shape[0]) * 100
perc_fraudes_em_A
print('Percentual de fraudes em A: {:.2f}%'.format(perc_fraudes_em_A
```

```
Percentual de fraudes em A: 0.17%
```

In [156]:
```
B['Class'].value_counts()
```

```
0    142157
1       246
Name: Class, dtype: int64
```

In [157]:
```
perc_fraudes_em_B = (B['Class'].value_counts()[1]/B.shape[0]) * 100
perc_fraudes_em_B
print('Percentual de fraudes em B: {:.2f}%'.format(perc_fraudes_em_B
```

```
Percentual de fraudes em B: 0.17%
```

In [158]:
```
def regra_fraude(x):
    if x >= 122.20:
        return 1
    else:
        return 0
```

In [165]:
```
B.head()
```

	Amount	Class	Pred_regra
163517	62.50	0	0
70537	3.57	0	0
9230	29.99	0	0
240317	550.00	0	1
93238	100.00	0	0

In [160]:
```
B['Pred_regra'] = B['Amount'].apply(regra_fraude)
```

In [163]:
```
print(classification_report(y_true=B['Class'], y_pred=B['Pred_regra'
```

```
                   precision    recall  f1-score   support

               0       1.00      0.83      0.91    142157
               1       0.00      0.26      0.01       246

        accuracy                           0.83    142403
       macro avg       0.50      0.54      0.46    142403
```

In [166]:
```
print(confusion_matrix(y_true=B['Class'], y_pred=B['Pred_regra']))
```

```
[[118429  23728]
 [   183     63]]
```

In [293]:
```
_ = B[B['Class'] == 1].shape[0]
print(f'Número de fraudes em B: {_}')
print('A regra aplicada classificou corretamente {:.2f}% das fraudes
```

```
Número de fraudes em B: 246
A regra aplicada classificou corretamente 0.26% das fraudes
```

5) Realize o seguinte teste de hipótese: transações frauduler gasto, na média, maiores ou iguais a 122.20

In [204]:
```
if (df[df['Class'] == 1]['Amount'].mean() - 122.2) >= 0:
    print('Hipótese verdadeira!')
else:
    print('Hipótese falsa!')
```

```
Hipótese verdadeira!
```

6) Utilizando as classes `LogisticRegression` e `SMOTE`, realize ove modelo, relatando qual a porcentagem de transações fraudul

In [254]:
```
df.head()
```

	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	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-(

5 rows × 31 columns

In [251]:
```
df.shape, df.columns
```

```
((284807, 31),
 Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
        'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
        'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
        'Class'],
       dtype='object'))
```

In [221]:
```
df['Class'].value_counts()
```

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

In [261]:
```
x = df.drop(labels=['Time', 'Class'], axis=1)
y = df['Class']

x.shape, y.shape
```

```
((284807, 29), (284807,))
```

In [262]:
```
oversample = SMOTE(random_state=666)
```

In [263]:
```
x_sm, y_sm = oversample.fit_resample(x, y)

x_sm.shape, y_sm.shape
```

```
((568630, 29), (568630,))
```

In [266]:
```
print(f'Porcentagem de dados de fraude: {100*sum(y_sm)/y_sm.shape[0]}
```

```
Porcentagem de dados de fraude: 50.0%
```

In [272]:
```
x_train, x_test, y_train, y_test = train_test_split(x_sm, y_sm, test_

x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((426472, 29), (426472,), (142158, 29), (142158,))
```

In [278]:
```
lr = LogisticRegression(max_iter=1e5)
```

In [279]:
```
lr.fit(x_train, y_train)
```

```
LogisticRegression(max_iter=10000.0)
```

In [281]:
```
pred = lr.predict(x_test)
```

In [282]:
```
print(classification_report(y_true=y_test, y_pred=pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96     71211
           1       0.98      0.94      0.96     70947

    accuracy                           0.96    142158
   macro avg       0.96      0.96      0.96    142158
weighted avg       0.96      0.96      0.96    142158
```

In [284]:
```
print(confusion_matrix(y_true=y_test, y_pred=pred))
```

```
[[69875  1336]
 [ 4471 66476]]
```

In [288]:
```
print('A regressão classificou corretamente {:.2f}% das fraudes'.for
```

```
A regressão classificou corretamente 98.03% das fraudes
```

In [ ]: