Importando libs

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

Importação de dados e primeiras infos

In [ ]:

```
# Carregando dados
df = pd.read_excel('https://github.com/silvacaio/Desafio_DataScience/raw/main/teste_smarkio_lbs.xls', sheet_name = 'NLP')
df.head()
```

Out[ ]:

	letra	artista
0	Jay-z Uh-uh-uh You ready b? Let's go get 'em. ...	Beyoncé
1	Your challengers are a young group from Housto...	Beyoncé
2	Dum-da-de-da Do, do, do, do, do, do (Coming do...	Beyoncé
3	If I ain't got nothing I got you If I ain't go...	Beyoncé
4	Six inch heels She walked in the club like nob...	Beyoncé

In [ ]:

```
#conhecendo a estrutura do dataset
df.shape
```

Out[ ]:

```
(518, 2)
```

In [ ]:

```
#tipo das colunas
df.dtypes
```

Out[ ]:

```
letra      object
artista    object
dtype: object
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 518 entries, 0 to 517
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   letra    518 non-null    object
 1   artista  518 non-null    object
dtypes: object(2)
memory usage: 8.2+ KB
```

Sem missing values, não precisamos fazer nenhuma trativa em cima disto.

EDA - Exploratory Data Analysis

usaremos o dataframe inicial, pois não iremos fazer nenhum ajustes

In [ ]:

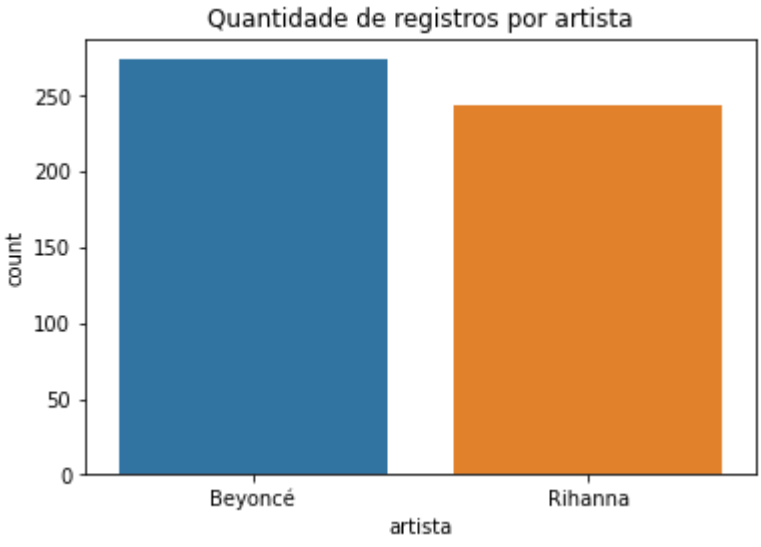
```
df_eda = df.copy()
```

In [ ]:

```
sns.countplot(data=df_eda, x='artista')
plt.title('Quantidade de registros por artista')
```

Out[ ]:

```
Text(0.5, 1.0, 'Quantidade de registros por artista')
```



Temos classes bem balanceadas!

In [ ]:

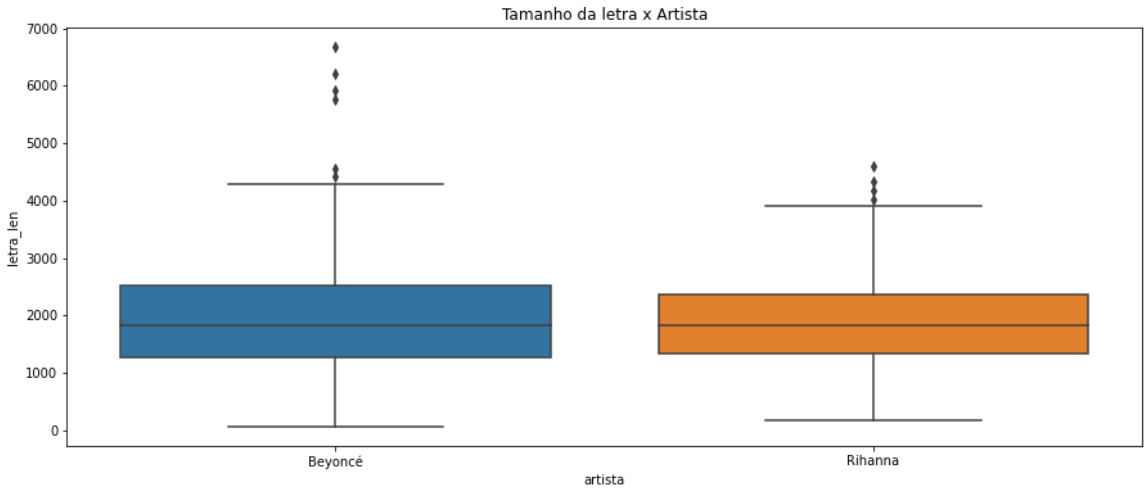
```
df_eda['letra_len'] = df_eda['letra'].str.len()
```

In [ ]:

```
plt.figure(figsize=(15, 6))
sns.boxplot(data=df_eda, x='artista', y='letra_len')
plt.title("Tamanho da letra x Artista")
```

Out[ ]:

```
Text(0.5, 1.0, 'Tamanho da letra x Artista')
```



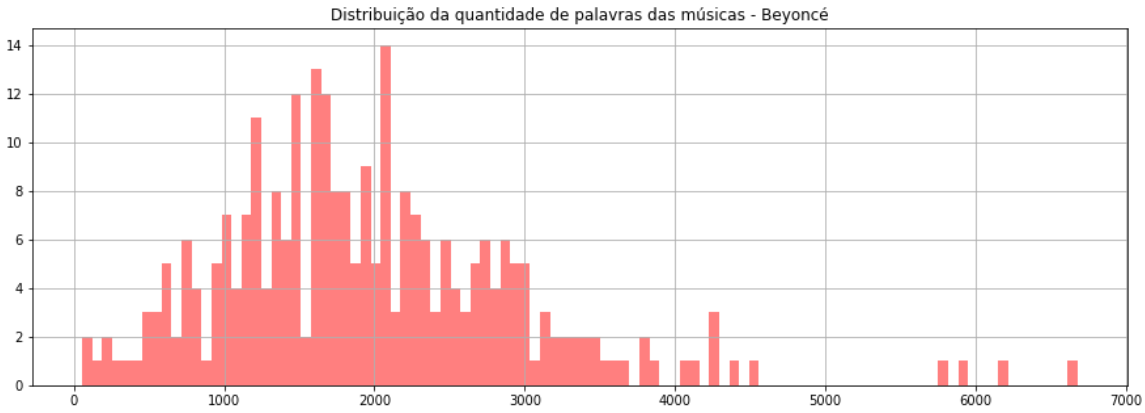
Possível observar que as duas artistas analisadas tem letras com tamanhos bem similares. Possível observar diversos outliers para a Beyoncé.

In [ ]:

```
df_eda[df_eda['artista'] == 'Beyoncé'].letra_len.hist(bins = 100, alpha = .5,color='red', figsize = (15, 5))
plt.title('Distribuição da quantidade de palavras das músicas - Beyoncé')
```

Out[ ]:

```
Text(0.5, 1.0, 'Distribuição da quantidade de palavras das músicas - Beyoncé')
```

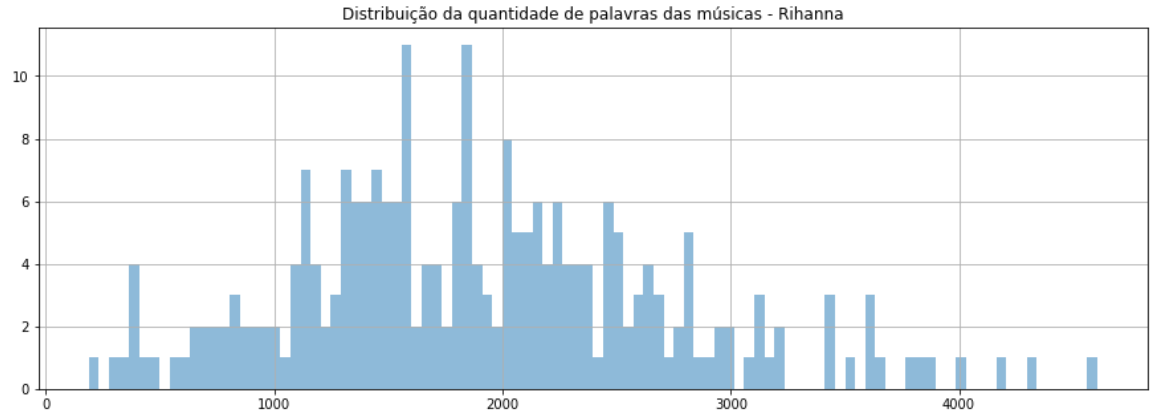


In [ ]:

```
df_eda[df_eda['artista'] == 'Rihanna'].letra_len.hist(bins = 100, alpha = .5, figsize =
(15, 5))
plt.title('Distribuição da quantidade de palavras das músicas - Rihanna')
```

Out[ ]:

```
Text(0.5, 1.0, 'Distribuição da quantidade de palavras das músicas - Rihan
na')
```



NLP

Conforme observado na análisa acima, temos alguns valores de outliers (principalmente para a classe Beyonce) e por isso iremos utilizar a tecnica TfidfTransformer (Term Frequency times inverse document frequency.)

In [ ]:

```
#Dados de treino e teste 80% e 20%
x_train, x_test, y_train, y_test = train_test_split(df['letra'], df['artista'], test_si
ze=0.2)
```

In [ ]:

```
#tokenização
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(stop_words='english')
```

In [ ]:

```
X_train_counts = count_vect.fit_transform(x_train)
X_test_counts = count_vect.transform(x_test)
```

In [ ]:

```
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_test_tfidf = tfidf_transformer.fit_transform(X_test_counts)
```

In [ ]:

```
#criação do modelo (naive bayes)
model = MultinomialNB().fit(X_train_tfidf, y_train)
```

In [ ]:

```
predictData = model.predict(X_test_tfidf)
```

In [ ]:

```
print(f"Acurácia: {round(accuracy_score(y_test,predictData)*100, 2)}%.")
```

```
Acurácia: 72.12%.
```