Unit 12 - Tales from the Crypto

1. Sentiment Analysis

Initially, you will use the sentiment scores for Bitcoin and Ethereum and create a DataFrame of sentiment scores for each coin.

Use descriptive statistics to answer the following questions:

1. Which coin has the highest mean positive score?
2. Which coin has the highest negative score?
3. Which coin has the highest positive score?

[1]:
```
import os
import pandas as pd
from dotenv import load_dotenv
load_dotenv()
from nltk.sentiment.vader import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
```

[2]:
```
# Read your api key environment variable
api_key = os.getenv("news_api")
api_key
```

[3]:
```
# Install the NewsAPI client
!pip install newsapi-python --upgrade
from newsapi.newsapi_client import NewsApiClient
```

[4]:
```
# Fetch the Bitcoin news articles
btc_sentiment = newsapi.get_everything(
    q="bitcoin",
    language="en",
    page_size=100,
    sort_by="relevancy"
)
```

[5]:
```
# Fetch the Ethereum news articles
eth_sentiment = newsapi.get_everything(q="ethereum", language="en", page_size=100, sort_by="relevancy")
len(eth_sentiment["articles"])
```

[6]:
```
# Create the Bitcoin sentiment scores DataFrame
btc_sentiment_df = pd.DataFrame(btc_sentiment["articles"])
btc_sentiment_df.head(5)
```

[7]:
```
# Create the Ethereum sentiment scores DataFrame
eth_sentiment_df = pd.DataFrame(eth_sentiment["articles"])
eth_sentiment_df.head(5)
```

[8]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[9]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[10]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[11]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[12]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[13]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[14]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[15]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[16]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[17]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[18]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[19]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[20]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[21]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[22]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[23]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[24]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[25]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[26]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[27]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[28]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[29]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[30]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[31]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[32]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[33]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[34]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[35]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[36]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[37]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[38]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[39]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[40]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[41]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[42]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[43]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[44]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[45]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[46]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[47]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[48]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[49]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[50]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[51]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[52]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[53]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[54]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[55]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[56]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[57]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[58]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[59]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[60]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[61]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[62]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[63]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[64]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[65]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[66]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[67]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[68]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[69]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[70]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[71]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[72]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[73]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[74]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[75]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[76]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[77]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[78]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[79]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[80]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[81]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[82]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[83]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[84]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[85]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[86]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[87]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[88]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[89]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[90]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[91]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[92]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[93]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[94]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[95]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[96]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[97]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[98]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[99]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[100]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[101]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[102]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[103]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[104]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[105]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[106]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[107]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[108]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[109]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[110]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[111]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[112]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[113]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[114]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[115]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[116]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[117]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[118]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[119]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[120]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[121]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[122]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[123]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[124]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[125]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[126]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[127]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[128]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[129]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[130]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[131]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[132]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[133]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[134]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[135]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[136]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[137]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[138]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[139]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[140]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[141]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[142]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[143]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[144]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[145]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[146]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_sentiment_df["content"].apply(sentiment)
btc_sentiment_df.head(5)
```

[147]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
eth_sentiment_df["sentiment"] = eth_sentiment_df["content"].apply(sentiment)
eth_sentiment_df.head(5)
```

[148]:
```
# Create a function to apply the VADER sentiment scoring
def sentiment(text):
    return analyzer.polarity_scores(text)
btc_sentiment_df["sentiment"] = btc_s
```

Bitcoin NER

Illustration: Tim Robinson | Reuters | Bitcoin price | Bitcoin | Ethereum | Cryptocurrency | Bitcoin | Bitcoin | Bitcoin

Ethereum NER

There are many Ethereum-specific computing for developers' attention right now. From the big-data side of Ethereum, Ethereum Network, to its native Ethereum Virtual Machine (EVM), Ethereum Network, Ethereum, Ethereum Virtual Machine (EVM) Ethereum Network is a decentralized blockchain platform that enables a decentralized, peer-to-peer network for executing peer-to-peer transactions.