In [1]:
```python
import pandas as pd

import re
from emoji import UNICODE_EMOJI
from textblob import TextBlob
import altair as alt
import numpy as np
from collections import Counter
import string

import nltk
nltk.download('vader_lexicon')
nltk.download('brown')
nltk.download('punkt')
nltk.download('stopwords')

from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/jovyan/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package brown to /home/jovyan/nltk_data...
[nltk_data]   Package brown is already up-to-date!
[nltk_data] Downloading package punkt to /home/jovyan/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/jovyan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

The data cleaning/manipulation functions

```
In [2]: def extract_tags(text):
            return re.findall("#([a-zA-Z0-9_]{1,50})", text)

        def extract_emoji(text):
            return [ch for ch in text if ch in UNICODE_EMOJI['en']]


        def clean_tweet(txt):
            temp = re.sub("@[A-Za-z0-9_]+","", txt)
            temp1 = re.sub("#[A-Za-z0-9_]+","", temp)
            temp2 = re.sub(r"http\S+", "", temp1)

            result=''.join(i for i in temp2.lower() if (i.isalpha() or i==' '))
            return result

        def word_list(tweet):

            lst = word_tokenize(tweet)
            lst1 = []
            stops = list(stopwords.words('english'))
            for w in lst:
                if w not in stops:
                    lst1.append(w)

            return lst1

        def sentiment(tweet):
            blob = TextBlob(tweet)

            return blob.sentiment.polarity


        def get_date(date):

            return date[:10]

        def get_hour(date):

            return date[11:13]
        def get_10min(date):

            return date[14]+'0'

        def get_min(date):
```

```
    return date[14:16]

def firm_pos(score):
    if score >= 0.7:
        return 1
    else: return 0

def pos(score):
    if (score >= 0.25) & (score < 0.7):
        return 1
    else: return 0

def neutral(score):
    if (score >= -0.25) & (score < 0.25):
        return 1
    else: return 0

def neg(score):
    if (score > -0.7) & (score < -0.25):
        return 1
    else: return 0

def firm_neg(score):
    if score <= -0.7:
        return 1
    else: return 0
```

Import data, check duplicate or missing value, remove rows if exists

In [3]:
```
df= pd.read_csv('Project Data/2020 nba finals.csv')
df['id'].duplicated(keep='last').sum()
```

Out[3]:
```
0
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
id      0
date    1
text    1
dtype: int64
```

In [5]:
```
df = df.dropna().reset_index()
df.drop(columns=['index'],inplace=True)
```

Apply data cleaning/manipulation techniques on the data, we now have the used words, tags, emojis, sentiment score, and specific date/hour/min data.

In [6]:
```python
df['tags']= df.apply(lambda row: extract_tags(row['text']), axis=1)
df['emojis']= df.apply(lambda row: extract_emoji(row['text']), axis=1)
df['clean_text']= df.apply(lambda row: clean_tweet(row['text']), axis=1)
df['words']= df.apply(lambda row: word_list(row['clean_text']), axis=1)
df['sentiment_score']= df.apply(lambda row: sentiment(row['clean_text']), axis=1)
df['day']= df.apply(lambda row: get_date(row['date']), axis=1)
df['hour']= df.apply(lambda row: get_hour(row['date']), axis=1)
df['10min']= df.apply(lambda row: get_10min(row['date']), axis=1)
df['min']= df.apply(lambda row: get_min(row['date']), axis=1)
df['POS']= df.apply(lambda row: firm_pos(row['sentiment_score']), axis=1)
df['pos']= df.apply(lambda row: pos(row['sentiment_score']), axis=1)
df['neu']= df.apply(lambda row: neutral(row['sentiment_score']), axis=1)
df['neg']= df.apply(lambda row: neg(row['sentiment_score']), axis=1)
df['NEG']= df.apply(lambda row: firm_neg(row['sentiment_score']), axis=1)

df.head()
```

Out[6]:

	id	date	text	tags	emojis	clean_text	words	sentiment_score	day	hour	10min	min	POS	p
0	1311455787101949952	2020-09-30 23:59:59+00:00	#QuestionOfTheDay \nWho will win game 1 of the...	[QuestionOfTheDay, NBAFinals, NBA, LeBronJames...	[]	who will win game of the or	[win, game]	0.20000	2020-09-30	23	50	59	0	
1	1311455786493792257	2020-09-30 23:59:59+00:00	Lets go @Lakers \n#NBAFinals \n#NBA	[NBAFinals, NBA]	[]	lets go	[lets, go]	0.00000	2020-09-30	23	50	59	0	
2	1311455784090234881	2020-09-30 23:59:59+00:00	Lebron is 1-8 in game 1 of NBA finals :(but c...	[]	[]	lebron is in game of nba finals but come on...	[lebron, game, nba, finals, come, boys]	-0.40000	2020-09-30	23	50	59	0	
3	1311455754566672384	2020-09-30 23:59:52+00:00	If you know you know #HEATCulture #Winning #NB...	[HEATCulture, Winning, NBAFinals]	[]	if you know you know	[know, know]	0.00000	2020-09-30	23	50	59	0	
4	1311455749168484353	2020-09-30 23:59:50+00:00	‼ NEW EPISODE ‼ \n\nEpisode 37 of the Open In...	[SoundCloud, NBAFinals]	[‼, ‼]	new episode episode of the open invitation p...	[new, episode, episode, open, invitation, podc...	0.12013	2020-09-30	23	50	59	0	

Overall tweet count & sentiment flow

In [7]:
```
tweet_count = df.groupby(['day','hour']).size().reset_index()
tweet_count['date'] = tweet_count['day'] + ' ' + tweet_count['hour'] + ':00'
tweet_count.columns = ['day', 'hour', 'count', 'date']

tweet_count.head()
```

Out[7]:

	day	hour	count	date
0	2020-09-30	00	404	2020-09-30 00:00
1	2020-09-30	01	425	2020-09-30 01:00
2	2020-09-30	02	445	2020-09-30 02:00
3	2020-09-30	03	320	2020-09-30 03:00
4	2020-09-30	04	275	2020-09-30 04:00

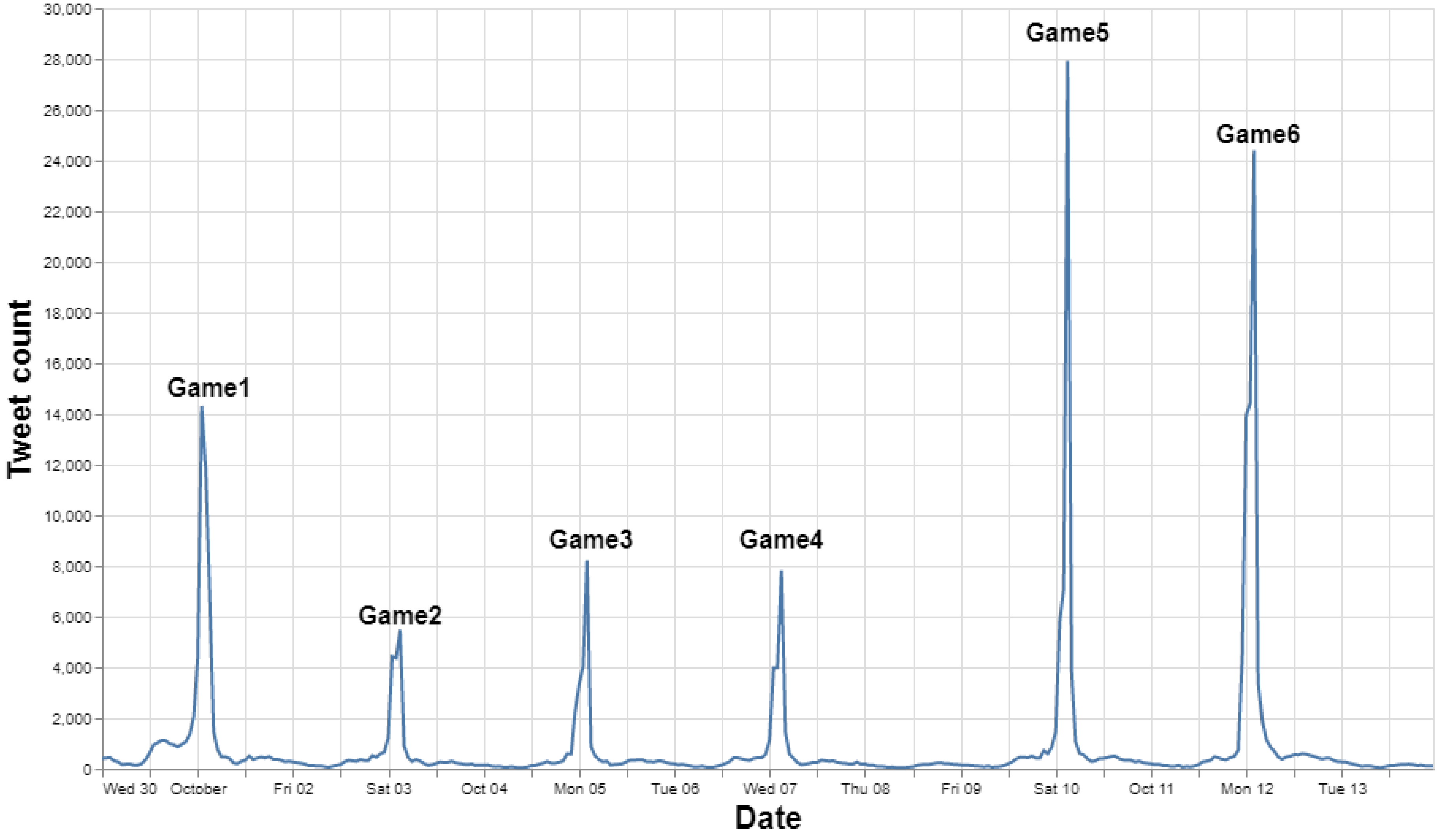
In [8]:

```python
annotations = [['2020-10-01 03:00',15000, 'Game1'],
               ['2020-10-03 03::00',6000, 'Game2'],
              ['2020-10-05 03::00',9000, 'Game3'],
               ['2020-10-07 03::00',9000, 'Game4'],
              ['2020-10-10 03::00',29000, 'Game5'],
              ['2020-10-12 03::00',25000, 'Game6']]
a_df = pd.DataFrame(annotations, columns=['date','values','note'])

line = alt.Chart(tweet_count).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('count:Q',title='Tweet count')
)

text=alt.Chart(a_df).encode(
    x=alt.X('date:T'),
    y=alt.Y('values:Q'),
    text='note').mark_text(size=16,fontWeight='bold')

(line + text).properties(width=840,height=480,title={
      "text": ["Tweets Count Flow - 2020 NBAFinals"],
      "subtitle": ["Tweet count about the Finals on Twitter by hour"]
    }).configure_axis(
    labelFontSize=10,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 28,
    subtitleFontSize = 15
)
```

Out[8]:

Tweets Count Flow - 2020 NBAFinals

Tweet count about the Finals on Twitter by hour



In [9]:

```python
# compute the 12-hour rolling average percentage of each 5 sentiment degree of tweets

senti = df.groupby(['day','hour']).sum()[['POS', 'pos', 'neu','neg', 'NEG']]
senti = senti.reset_index()
senti['date'] = senti['day'] + ' ' + senti['hour'] + ':00'
senti['size'] = pd.Series(df.groupby(['day','hour']).size().values)
senti[['POS', 'pos', 'neu', 'neg', 'NEG','12hr_count']] = senti.rolling(window=12,min_periods=1).sum()[['POS', 'pos', 'neu', 'neg'
senti['POSITIVE'] = senti['POS'] / senti['12hr_count']
senti['positive'] = senti['pos'] / senti['12hr_count']
senti['neutral'] = senti['neu'] / senti['12hr_count']
senti['negative'] = senti['neg'] / senti['12hr_count']
senti['NEGATIVE'] = senti['NEG'] / senti['12hr_count']

senti.head()
```

Out[9]:

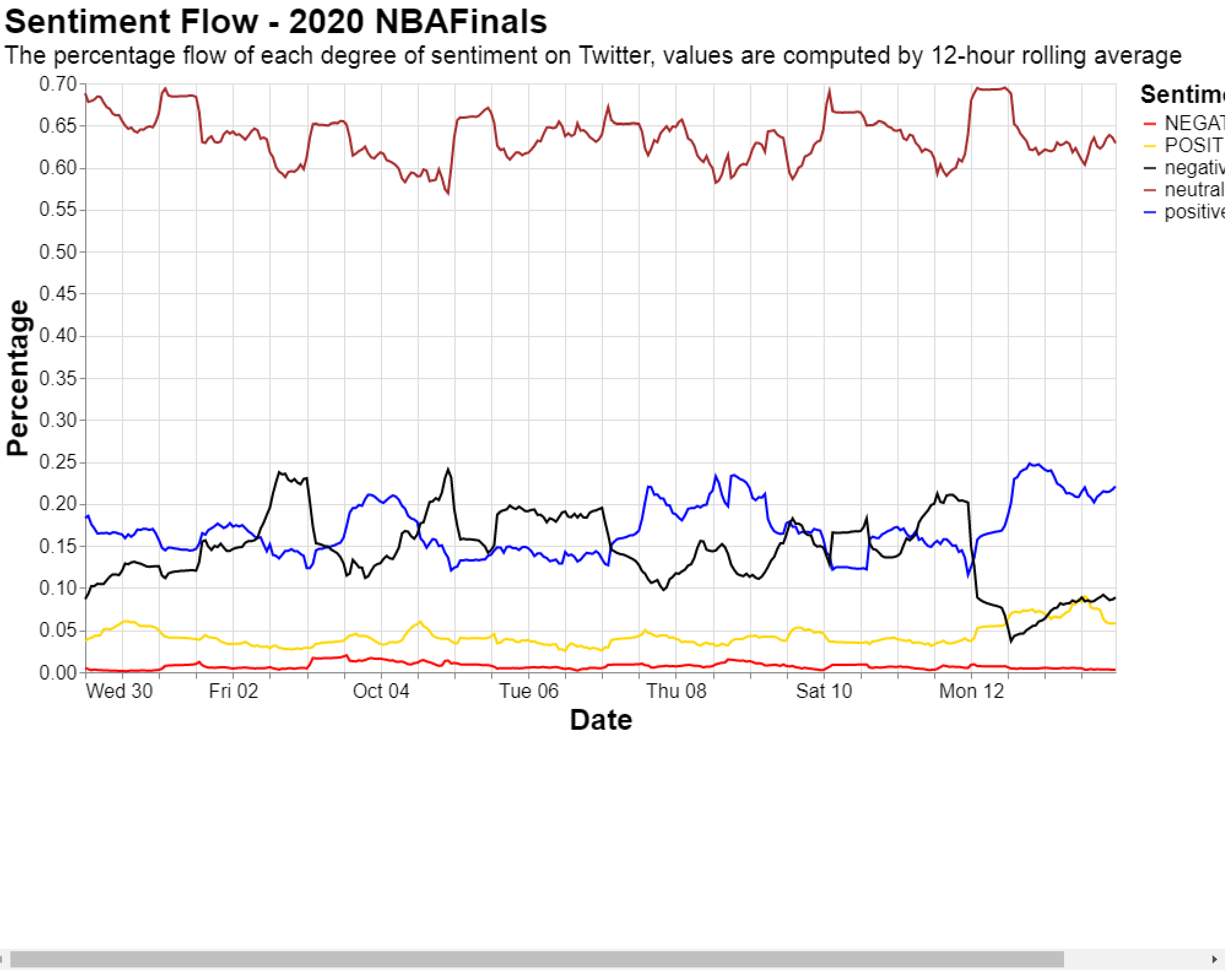
	day	hour	POS	pos	neu	neg	NEG	date	size	12hr_count	POSITIVE	positive	neutral	negative	NEGATIVE
0	2020-09-30	00	15.0	74.0	278.0	35.0	2.0	2020-09-30 00:00	404	404.0	0.037129	0.183168	0.688119	0.086634	0.004950
1	2020-09-30	01	33.0	154.0	562.0	77.0	3.0	2020-09-30 01:00	425	829.0	0.039807	0.185766	0.677925	0.092883	0.003619
2	2020-09-30	02	52.0	224.0	865.0	130.0	3.0	2020-09-30 02:00	445	1274.0	0.040816	0.175824	0.678964	0.102041	0.002355
3	2020-09-30	03	69.0	272.0	1085.0	163.0	5.0	2020-09-30 03:00	320	1594.0	0.043287	0.170640	0.680678	0.102258	0.003137
4	2020-09-30	04	81.0	308.0	1279.0	196.0	5.0	2020-09-30 04:00	275	1869.0	0.043339	0.164794	0.684323	0.104869	0.002675

In [10]:
```
# create a dataframe for visualization

date = []
value = []
label = []

senti_flow = pd.DataFrame()

for i in ['POSITIVE', 'positive', 'neutral', 'negative', 'NEGATIVE']:
    lst=[]
    lst1=[]
    lst2=list(senti.date.values)

    for j in range(len(senti)):
        lst.append(i)
        lst1.append(senti[i][j])

    date += lst2
    value += lst1
    label += lst

senti_flow['date'] = pd.Series(date)
senti_flow['Sentiment_label'] = pd.Series(label)
senti_flow['perct'] = pd.Series(value)
senti_flow.head()
```

Out[10]:

	date	Sentiment_label	perct
0	2020-09-30 00:00	POSITIVE	0.037129
1	2020-09-30 01:00	POSITIVE	0.039807
2	2020-09-30 02:00	POSITIVE	0.040816
3	2020-09-30 03:00	POSITIVE	0.043287
4	2020-09-30 04:00	POSITIVE	0.043339

In [11]:
```
alt.Chart(senti_flow).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('perct:Q',title='Percentage'),
    color=alt.Color('Sentiment_label',
                   scale=alt.Scale(
            range=['red', 'gold','black','brown','blue']))
).properties(width=840,height=480,title={
      "text": ["Sentiment Flow - 2020 NBAFinals"],
      "subtitle": ["The percentage flow of each degree of sentiment on Twitter, values are computed by 12-hour rolling average"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=24
).configure_title(
    anchor='start',
    fontSize = 28,
    subtitleFontSize = 20
).configure_legend(
    titleFontSize=20,
    labelFontSize=16
)
```
Out[11]:


Sentiment Flow - 2020 NBAFinals
The percentage flow of each degree of sentiment on Twitter, values are computed by 12-hour rolling average
Percentage
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Wed 30
Fri 02
Oct 04
Tue 06
Thu 08
Sat 10
Mon 12
Date
Sentim
NEGAT
POSIT
negativ
neutral
positive

Emoji/tag/word

```
In [12]: # this return the top 50 most common items in the columns (emoji/tag/word)

         def top_item(data,label):

             lst = []
             for i in data[label]:
                 lst += i

             C = Counter(lst)
             top50 = C.most_common(50)
             count_df = pd.DataFrame(top50,columns = [label,'count'])

             return count_df
```

In [13]:
```
c= top_item(df,'tags')
c

# the top 50 most frequently used tags within the 2020 nba finals tweets
```

Out[13]:

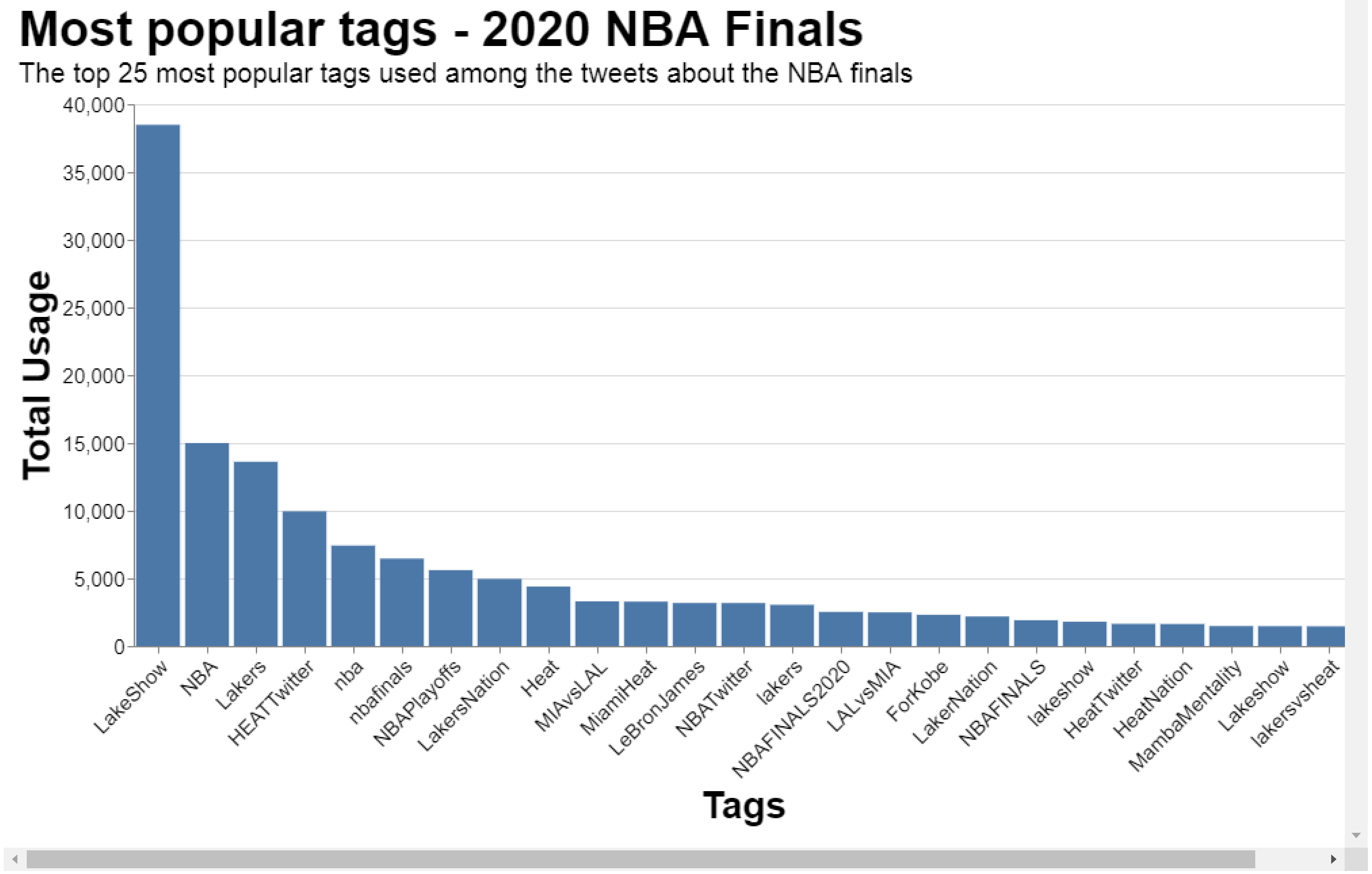
	tags	count
0	NBAFinals	202711
1	LakeShow	38474
2	NBA	15002
3	Lakers	13621
4	HEATTwitter	9958
5	nba	7427
6	nbafinals	6466
7	NBAPlayoffs	5610
8	LakersNation	4962
9	Heat	4395
10	MIAvsLAL	3302
11	MiamiHeat	3282
12	LeBronJames	3182
13	NBATwitter	3179
14	lakers	3047
15	NBAFINALS2020	2524
16	LALvsMIA	2487
17	ForKobe	2312
18	LakerNation	2183
19	NBAFINALS	1909
20	lakeshow	1797
21	HeatTwitter	1643
22	HeatNation	1629
23	MambaMentality	1477
24	Lakeshow	1468

	tags	count
25	lakersvsheat	1457
26	WholeNewGame	1407
27	NFL	1372
28	NBAChamps	1272
29	NBAfinals	1129
30	heat	1098
31	KingJames	1037
32	MambaForever	965
33	basketball	959
34	LeBron	957
35	NBA2K21	900
36	HEATCulture	889
37	NBAnaESPN	882
38	Kobe	858
39	MLBPlayoffs	844
40	17	805
41	LAKERSNATION	773
42	LosAngelesLakers	764
43	JimmyButler	738
44	Nba	701
45	NBATwitterLive	694
46	KobeBryant	694
47	2	686
48	AnthonyDavis	675
49	LakersVsHeat	654

Visualize bar chart for the most frequent tags

In [14]:
```
c1=c[1:26]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('tags',sort=['count'],title='Tags',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular tags - 2020 NBA Finals"],
      "subtitle":["The top 25 most popular tags used among the tweets about the NBA finals"]
    }).configure_axis(
    labelFontSize=15,
    titleFontSize=28
).configure_title(
    anchor='start',
    fontSize = 36,
    subtitleFontSize = 20
)
```

Out[14]:


Most popular tags - 2020 NBA Finals
The top 25 most popular tags used among the tweets about the NBA finals
Total Usage
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
LakeShow
NBA
Lakers
HEATTwitter
nba
nbafinals
NBAPlayoffs
LakersNation
Heat
MIAvsLAL
MiamiHeat
LeBronJames
NBATwitter
lakers
NBAFINALS2020
LALvsMIA
ForKobe
LakerNation
NBAFINALS
lakeshow
HeatTwitter
HeatNation
MambaMentality
Lakeshow
lakersvsheat
Tags

Emoji

In [15]:
```
c= top_item(df,'emojis')
c
```

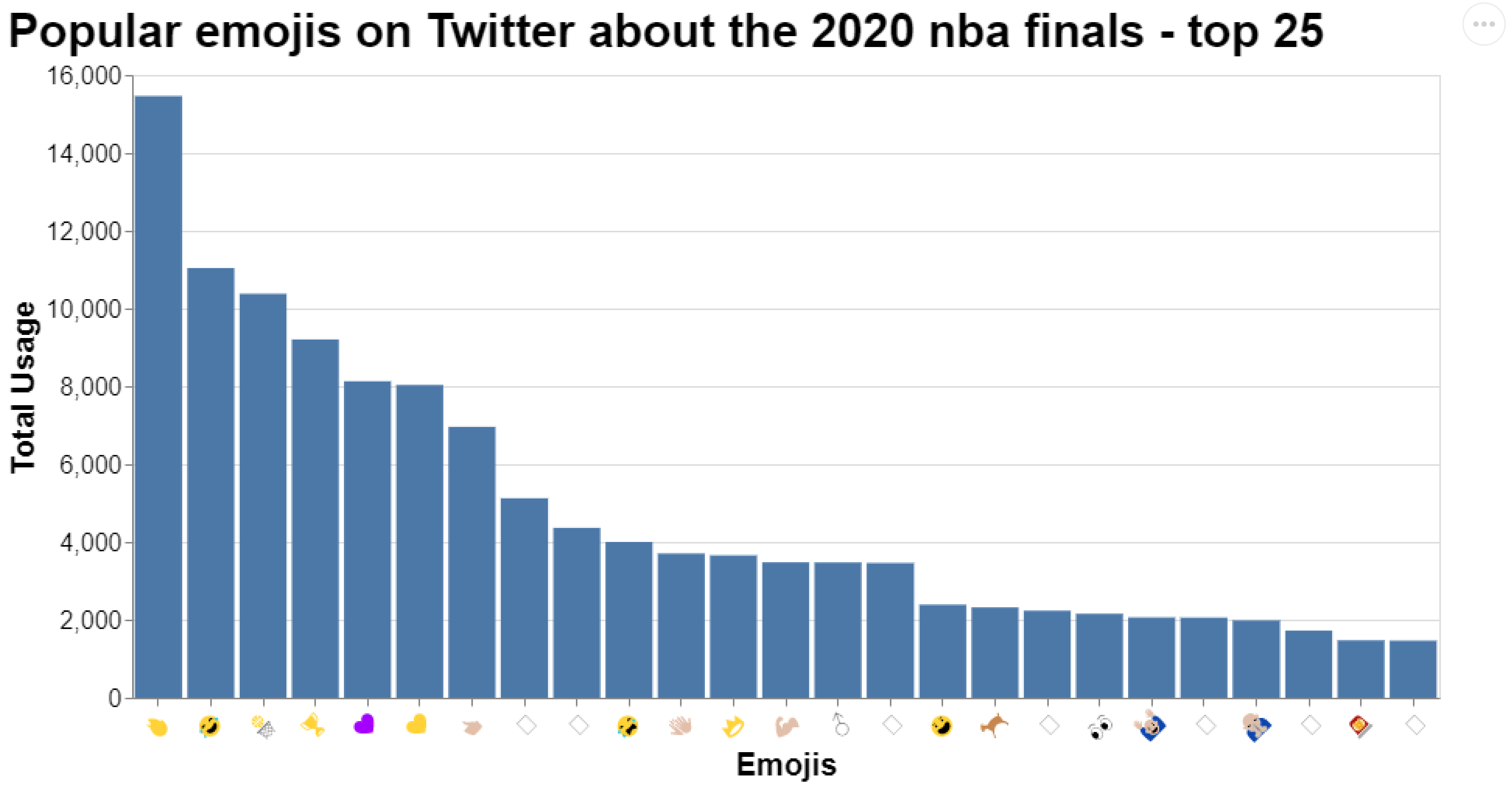
Out[15]:

	emojis	count
0	🔥	15459
1	😂	11041
2	🏀	10379
3	🏆	9203
4	🖤	8127
5	💜	8031
6	👍	6964
7		5126
8		4364
9	😭	4001
10	🙌	3704
11	👑	3655
12	💪	3476
13	♂	3472
14		3459
15	😃	2386
16	🐐	2316
17		2231
18	👀	2152
19	🙌	2058
20		2053
21	🙏	1983
22		1718
23	🚨	1469
24		1460

	emojis	count
25		1454
26	💯	1422
27	‼	1381
28	❤	1365
29	👇	1344
30		991
31	✔	975
32	😍	973
33		962
34	🐍	929
35	🎉	900
36	💰	860
37	🖥	781
38	😂	719
39	🤤	699
40		693
41	💍	676
42		666
43	😎	648
44		637
45	👌	624
46	♀	623
47		610
48	😟	594
49		580

In [16]:
```python
c1=c[:25]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('emojis',sort=['count'],title='Emojis',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=840,height=400,title = 'Popular emojis on Twitter about the 2020 nba finals - top 25'
).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 30
)
```

Out[16]:



Target specific time period for detailed analysis: Game6 (Laker seize championship)

In [17]:
```python
df['Date'] = pd.to_datetime(df['date'])
mask = (df['Date'] > '2020-10-11 22:00') & (df['Date'] < '2020-10-12 06:00')
game6 = df.loc[mask].sort_values('Date')
game6 = game6.reset_index()
game6.drop(columns=['index','Date'],inplace=True)

game6.head()
```

Out[17]:

	id	date	text	tags	emojis	clean_text	words	sentiment_score	day	hour	10min	min	POS	pos	neu	neg
0	1315411873265745921	2020-10-11 22:00:04+00:00	2020 NBA Finals Game 6 Open Thread\n\nIt start...	[]	[]	nba finals game open threadit starts at pm ...	[nba, finals, game, open, threadit, starts, pm...	-0.200	2020-10-11	22	00	00	0	0	1	0
1	1315411905058689024	2020-10-11 22:00:11+00:00	The Lakers' Alex Caruso will get his first sta...	[]	[]	the lakers alex caruso will get his first star...	[lakers, alex, caruso, get, first, start, nba,...	0.375	2020-10-11	22	00	00	0	1	0	0
2	1315411926361538560	2020-10-11 22:00:17+00:00	Can someone stream the NBA finals for me pleas...	[]	[]	can someone stream the nba finals for me pleas...	[someone, stream, nba, finals, please, wna, wa...	0.000	2020-10-11	22	00	00	0	0	1	0
3	1315411941515497473	2020-10-11 22:00:20+00:00	NBA Finals continues TONIGHT at 7:30 PM EST......	[kobe]	[💜, 💛, 🏀]	nba finals continues tonight at pm est lakers...	[nba, finals, continues, tonight, pm, est, lak...	0.800	2020-10-11	22	00	00	1	0	0	0
4	1315411972926705665	2020-10-11 22:00:28+00:00	The Lake Show or the Heat? #NBAFinals \nGame 6...	[NBAFinals]	[🏀]	the lake show or the heat game tonight at p...	[lake, show, heat, game, tonight, pm, et, abc]	-0.400	2020-10-11	22	00	00	0	0	0	1

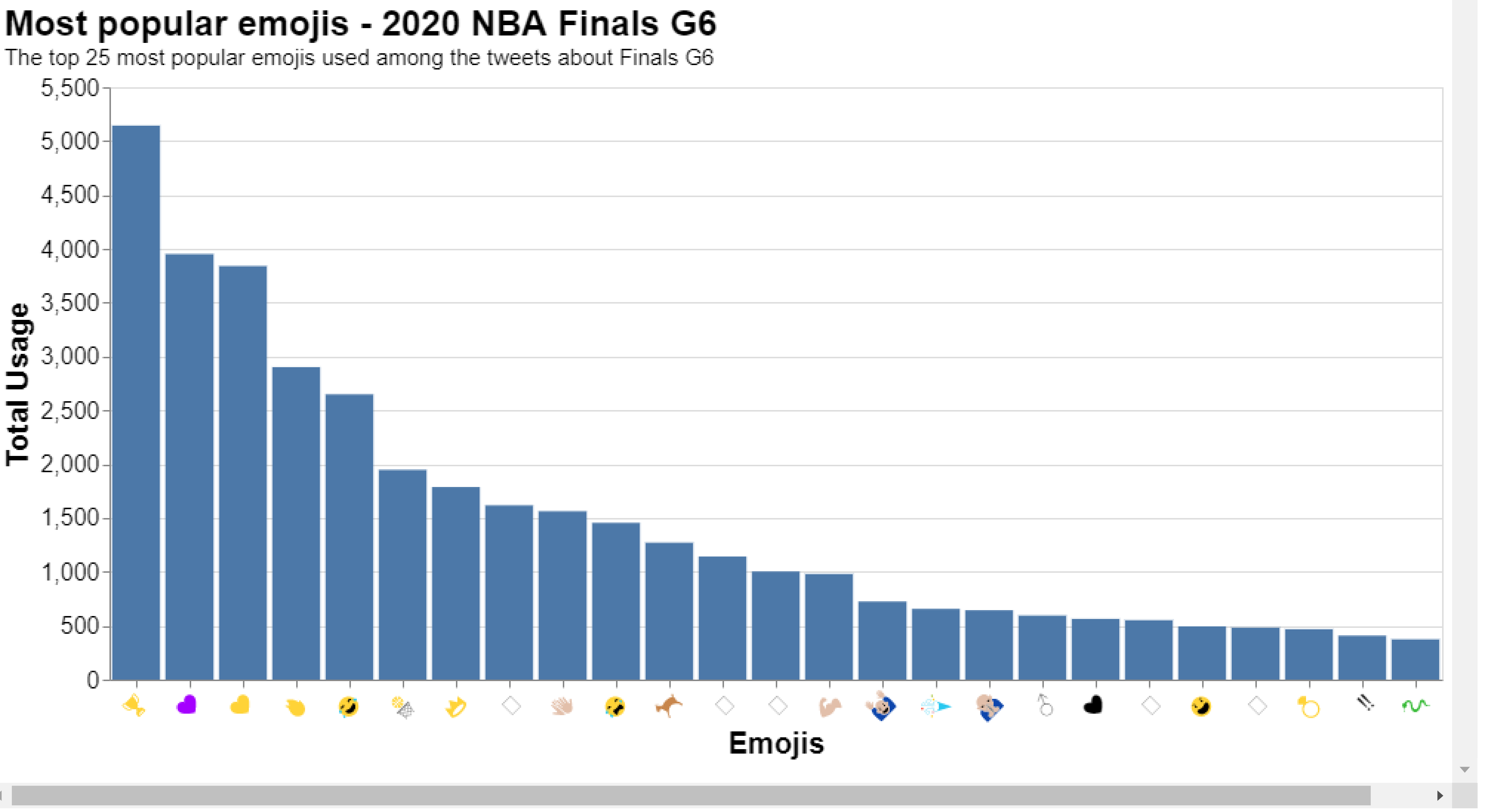
```
In [18]: count = top_item(game6,'emojis')
         count.head(20)
```

```
Out[18]:
```

	emojis	count
0	🏆	5144
1	💜	3953
2	💛	3842
3	🔥	2902
4	😂	2649
5	🏀	1947
6	👑	1787
7		1617
8	🙌	1562
9	😭	1454
10	🐐	1270
11		1141
12		1002
13	💪	979
14	🙌	726
15	🎉	658
16	🙏	645
17	♂	596
18	❤	565
19		553

In [19]:
```
c1=count[:25]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('emojis',sort=['count'],title='Emojis',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular emojis - 2020 NBA Finals G6"],
      "subtitle":["The top 25 most popular emojis used among the tweets about Finals G6"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[19]:


Most popular emojis - 2020 NBA Finals G6
The top 25 most popular emojis used among the tweets about Finals G6
Total Usage
5,500
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
Emojis

In [20]:
```python
c= top_item(game6,'tags')
c
```

Out[20]:

	tags	count
0	NBAFinals	53667
1	LakeShow	12494
2	Lakers	3804
3	NBA	3558
4	LakersNation	3230
5	nba	1719
6	nbafinals	1311
7	LeBronJames	1275
8	NBAPlayoffs	1109
9	ForKobe	1090
10	lakers	777
11	NBAChamps	763
12	HEATTwitter	722
13	LakerNation	676
14	MambaMentality	639
15	NBATwitter	612
16	NBAFINALS2020	543
17	Heat	542
18	MiamiHeat	534
19	MambaForever	531
20	lakeshow	473
21	LALvsMIA	446
22	LakersVsHeat	437
23	KingJames	417
24	17	408
25	Lakeshow	396

	tags	count
26	HeatNation	391
27	NBAFINALS	366
28	KobeBryant	347
29	Kobe	293
30	HeatTwitter	282
31	MVP	279
32	GOAT	259
33	LeBron	259
34	NBAChampions	255
35	LosAngelesLakers	240
36	NBAfinals	238
37	WholeNewGame	233
38	Game6	230
39	Champions	212
40	ForKobeAndGigi	190
41	SARSMUSTEND	173
42	Nba	169
43	AnthonyDavis	166
44	heat	165
45	LALakers	160
46	Kobethisisforyou	156
47	LakersvsMiami	154
48	NBAnaESPN	152
49	LosAngeles	145

Plot sentiment Flow in game6

In [21]:

```python
# compute the 15-minute rolling average percentage of each 5 sentiment degree

senti = game6.groupby(['day','hour','min']).sum()[['POS', 'pos', 'neu','neg', 'NEG']]
senti = senti.reset_index()
senti['date'] = senti['day'] + ' ' + senti['hour'] + ':' + senti['min']
senti['size'] = pd.Series(game6.groupby(['day','hour','min']).size().values)
senti[['POS', 'pos', 'neu', 'neg', 'NEG','15m_count']] = senti.rolling(window=15,min_periods=1).sum()[['POS', 'pos', 'neu', 'neg',
senti['POSITIVE'] = senti['POS'] / senti['15m_count']
senti['positive'] = senti['pos'] / senti['15m_count']
senti['neutral'] = senti['neu'] / senti['15m_count']
senti['negative'] = senti['neg'] / senti['15m_count']
senti['NEGATIVE'] = senti['NEG'] / senti['15m_count']

senti.head()
```

Out[21]:

	day	hour	min	POS	pos	neu	neg	NEG	date	size	15m_count	POSITIVE	positive	neutral	negative	NEGATIVE
0	2020-10-11	22	00	1.0	1.0	2.0	2.0	0.0	2020-10-11 22:00	6	6.0	0.166667	0.166667	0.333333	0.333333	0.0
1	2020-10-11	22	01	1.0	1.0	8.0	7.0	0.0	2020-10-11 22:01	11	17.0	0.058824	0.058824	0.470588	0.411765	0.0
2	2020-10-11	22	02	1.0	3.0	17.0	9.0	0.0	2020-10-11 22:02	13	30.0	0.033333	0.100000	0.566667	0.300000	0.0
3	2020-10-11	22	03	1.0	3.0	23.0	10.0	0.0	2020-10-11 22:03	7	37.0	0.027027	0.081081	0.621622	0.270270	0.0
4	2020-10-11	22	04	1.0	4.0	35.0	12.0	0.0	2020-10-11 22:04	15	52.0	0.019231	0.076923	0.673077	0.230769	0.0

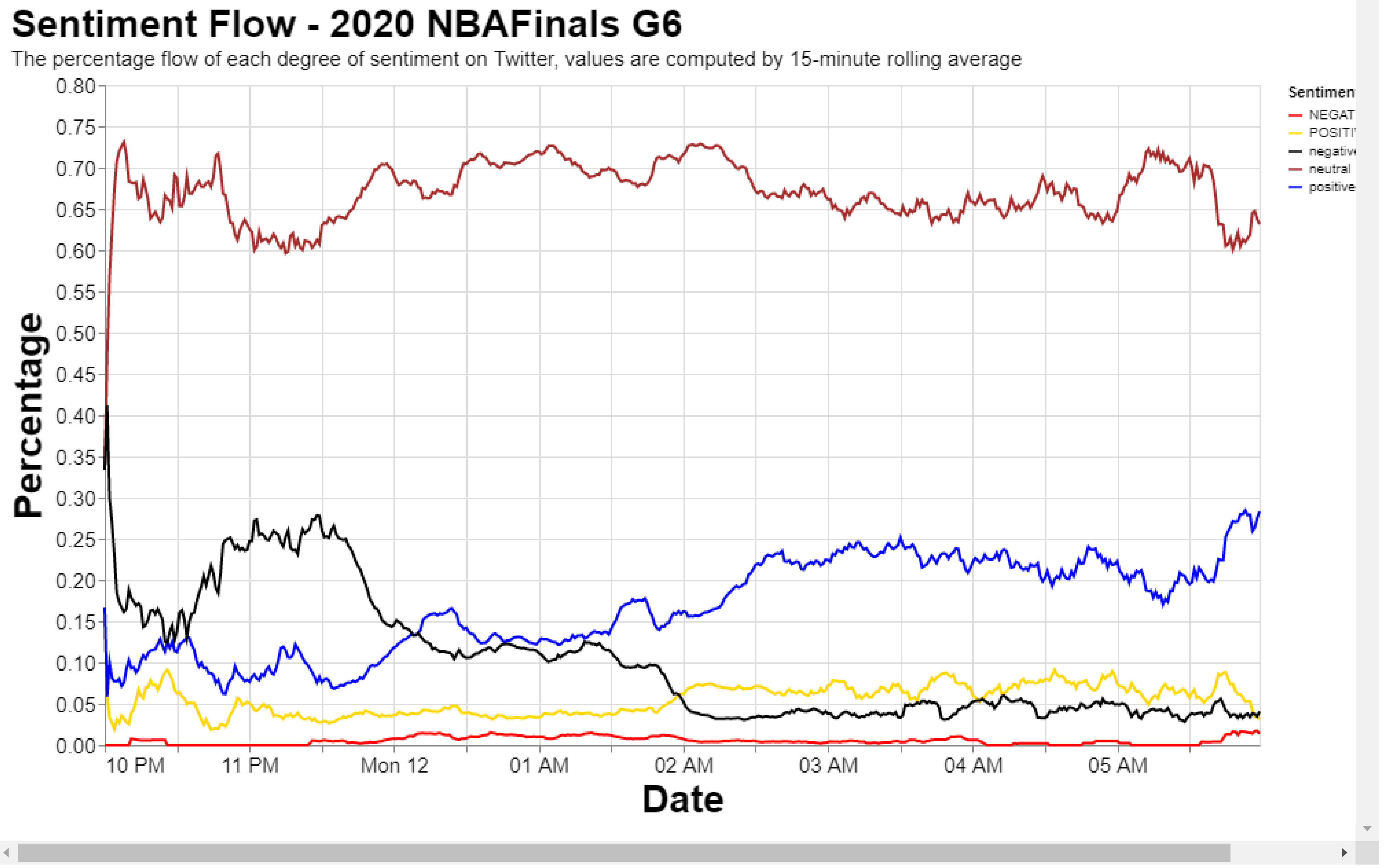
In [22]:
```
date = []
value = []
label = []

senti_flow = pd.DataFrame()

for i in ['POSITIVE', 'positive', 'neutral', 'negative', 'NEGATIVE']:
    lst=[]
    lst1=[]
    lst2=list(senti.date.values)

    for j in range(len(senti)):
        lst.append(i)
        lst1.append(senti[i][j])

    date += lst2
    value += lst1
    label += lst

senti_flow['date'] = pd.Series(date)
senti_flow['Sentiment_label'] = pd.Series(label)
senti_flow['perct'] = pd.Series(value)
senti_flow.head()
```

Out[22]:

	date	Sentiment_label	perct
0	2020-10-11 22:00	POSITIVE	0.166667
1	2020-10-11 22:01	POSITIVE	0.058824
2	2020-10-11 22:02	POSITIVE	0.033333
3	2020-10-11 22:03	POSITIVE	0.027027
4	2020-10-11 22:04	POSITIVE	0.019231

In [23]:
```python
alt.Chart(senti_flow).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('perct:Q',title='Percentage'),
    color=alt.Color('Sentiment_label',
                   scale=alt.Scale(
            range=['red', 'gold','black','brown','blue']))
).properties(width=840,height=480,title={
      "text": ["Sentiment Flow - 2020 NBAFinals G6"],
      "subtitle": ["The percentage flow of each degree of sentiment on Twitter, values are computed by 15-minute rolling average"]
    }).configure_axis(
    labelFontSize=15,
    titleFontSize=28
).configure_title(
    anchor='start',
    fontSize = 28,
    subtitleFontSize = 15
)
```

Out[23]:


Sentiment Flow - 2020 NBAFinals G6
The percentage flow of each degree of sentiment on Twitter, values are computed by 15-minute rolling average
Percentage
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
10 PM
11 PM
Mon 12
01 AM
02 AM
03 AM
04 AM
05 AM
Date
Sentimen
NEGAT
POSITI
negative
neutral
positive

```
In [ ]:
```