In [5]:
```
# Libraries to help with reading and manipulating data
import numpy as np
import pandas as pd

# Libraries to help with data visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
# Library to help with statistical analysis
import scipy.stats as stats
# import the scipy package and check the version to be sure that the v
import scipy
#read the dataset
df=pd. read_csv('Downloads/az_tunes.csv')
df
```

Out[5]:

	user_id	age_group	subscription_status	engagement_time
0	14451	18-34	subscribed	5.55
1	18386	under 18	subscribed	5.12
2	12305	35 and over	not_subscribed	4.25
3	17546	18-34	subscribed	8.54
4	15399	18-34	subscribed	12.12
...	...	...	...	...
995	17439	35 and over	subscribed	3.12
996	10112	35 and over	subscribed	5.25
997	10692	35 and over	not_subscribed	2.37
998	11164	18-34	subscribed	8.19
999	14958	35 and over	not_subscribed	3.78

1000 rows × 4 columns

In [6]:
```
df.head()
```

Out[6]:

	user_id	age_group	subscription_status	engagement_time
0	14451	18-34	subscribed	5.55
1	18386	under 18	subscribed	5.12
2	12305	35 and over	not_subscribed	4.25
3	17546	18-34	subscribed	8.54
4	15399	18-34	subscribed	12.12

In [11]:
```
df.shape
```

Out[11]: (1000, 4)

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   user_id              1000 non-null   int64  
 1   age_group            1000 non-null   object 
 2   subscription_status  1000 non-null   object 
 3   engagement_time      1000 non-null   float64
dtypes: float64(1), int64(1), object(2)
memory usage: 31.4+ KB
```

In [10]:
```
df.isna().sum()
```

Out[10]:
```
user_id                0
age_group              0
subscription_status    0
engagement_time        0
dtype: int64
```

In [12]:
```
df.describe()
```

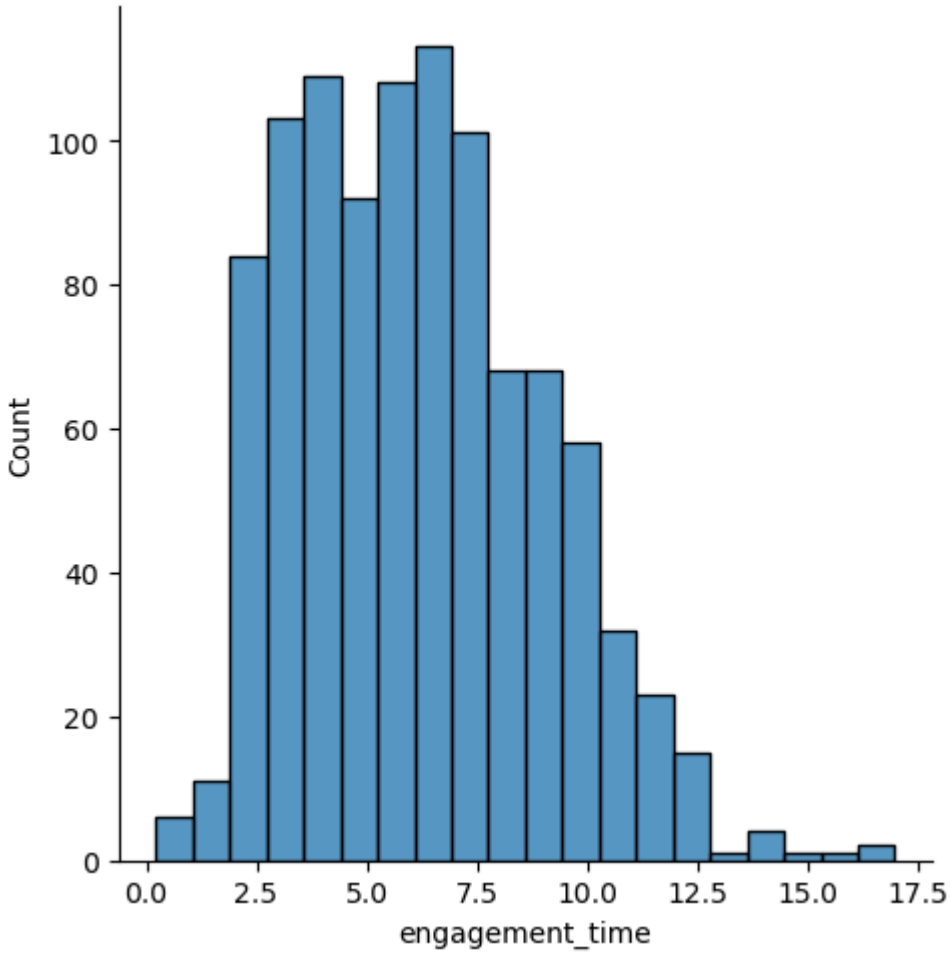
Out[12]:

	user_id	engagement_time
count	1000.000000	1000.000000
mean	15024.803000	6.180030
std	2927.044957	2.757166
min	10000.000000	0.220000
25%	12452.500000	3.917500
50%	15184.000000	6.000000
75%	17481.250000	8.110000
max	19976.000000	16.980000

In [14]:
```
# plot the distribution plot of engagement-time
print ('Sample mean:', np.round(df .engagement_time.mean (),2))
```

```
Sample mean: 6.18
```

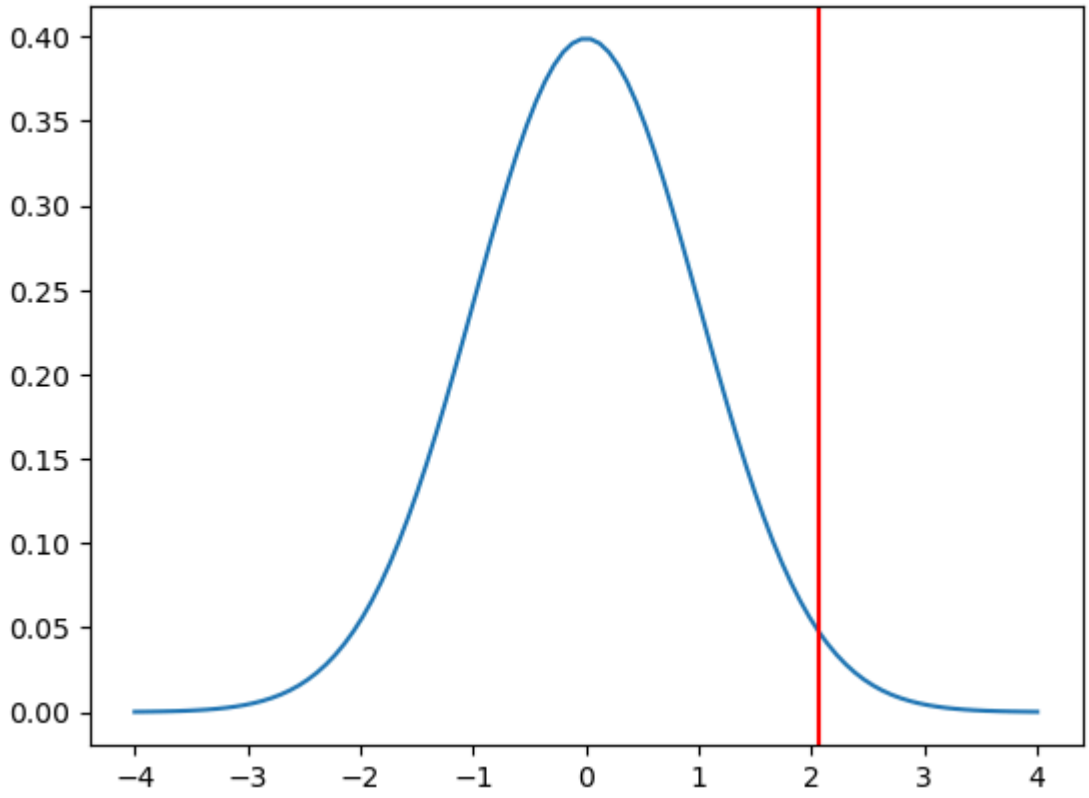
In [15]:
```
sns.displot(df.engagement_time)
plt.show()
```



In [19]:
```
hyp_mean = 6
t_stat, p_value = stats.ttest_1samp(df['engagement_time'], hyp_mean, a
print("Test Statistic=",t_stat)
print ("p-value =",p_value)
```

```
Test Statistic= 2.0648187232381248
p-value = 0.019598877431817586
```

In [21]:
```
#rio re testar distribution
# import the required function
from scipy.stats import t
# plotting the distribution of t test statistic along with the compute
# We are plotting the distributions here to better visualize the calcu
x = np.linspace(-4, 4, 100) # create an array of 100 numbers starting
plt.plot(x, t.pdf(x,df=len (df)-1)) # plot the pdf of the t distributi
plt.axvline(x = t_stat, c = 'r') # draw a vertical red line through th
plt.show() # display the plot
```



In [24]:
```
# print the conclusion based on p-value
if p_value < 0.05:
    print (f'As the p-value {p_value} is less than the level of signi
else:
    print(f'As the p-value {p_values} is greater than the level of sig
```

```
As the p-value 0.019598877431817586 is less than the level of signif
icance
```

In [25]:
```
# prepare a contingency table to perform the test
contingency_table= pd.crosstab(df.age_group,df.subscription_status)
contingency_table
```

Out[25]:

subscription_status	not_subscribed	subscribed
age_group		
18-34	103	262
35 and over	237	171
under 18	107	120

In [31]:
```
from scipy.stats import chi2_contingency

# Assume `contingency_table` is your input data
# For example: contingency_table = [[observed_row1], [observed_row2],

# Perform the chi-square test
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table

# Print the results
print("Chi-Square Test Statistic =", chi2_stat)
print("p-value =", p_value)
print("Degrees of Freedom =", dof)
print("Expected Frequencies:\n", expected)

# Interpret the results
if p_value < 0.05:
    print("Conclusion: There is a significant association between the
else:
    print("Conclusion: There is no significant association between the
```

```
Chi-Square Test Statistic = 70.23716243606756
p-value = 5.600076564450542e-16
Degrees of Freedom = 2
Expected Frequencies:
 [[163.155 201.845]
 [182.376 225.624]
 [101.469 125.531]]
Conclusion: There is a significant association between the variable
s.
```

In [ ]: