In [2]:
```python
#import necessary packages
import pandas as pd
```

In [3]:
```python
#import the data
df = pd.read_csv(r'C:\Users\ahzha\Desktop\TGG_Data_Takehome_(3).csv')
```

In [4]:
```python
#view the data
df.head(2)
```

Out[4]:

	trip_id	variable	value
0	37892018043022592800	start_date	2018-04-30 22:59:28 UTC
1	22302018043022173200	start_date	2018-04-30 22:17:32 UTC

In [5]:
```python
#Calculate rows & columns
df.shape
```

Out[5]:
```
(130000, 3)
```

In [6]:
```python
#Calculate missing values
df.isnull().sum()
```

Out[6]:
```
trip_id         0
variable        0
value       18670
dtype: int64
```

In [7]:
```python
#find all the variables names
variables = df.variable.unique()
```

In [8]:
```python
variables
```

Out[8]:
```
array(['start_date', 'start_station_name', 'start_station_id', 'end_date',
       'end_station_name', 'end_station_id', 'bike_number', 'zip_code',
       'subscriber_type', 'c_subscription_type', 'member_birth_year',
       'member_gender', 'bike_share_for_all_trip'], dtype=object)
```

In [9]:
```python
#create index and transform variable rows to columns
df1 = df.pivot(index='trip_id', columns='variable', values='value')
```

In [10]:
```python
print(df1)
```

```
variable             bike_number bike_share_for_all_trip c_subscription_type  \
trip_id
10002017102810135000        1000                     NaN          Subscriber
10002017112118254500        1000                     NaN          Subscriber
10002018020418322600        1000                      No          Subscriber
10012017112918483500        1001                     NaN            Customer
10012017121415293700        1001                     NaN          Subscriber
...                          ...                     ...                 ...
9982017103011564900          998                     NaN          Subscriber
9982017112108301300          998                     NaN          Subscriber
9982018012320471300          998                      No          Subscriber
9992017082616550000          999                     NaN            Customer
9992018010308503000          999                      No          Subscriber

variable                             end_date end_station_id  \
trip_id
10002017102810135000  2017-10-28 10:27:54 UTC            169
10002017112118254500  2017-11-21 18:33:46 UTC            200
10002018020418322600  2018-02-04 18:35:29 UTC            196
10012017112918483500  2017-11-29 18:58:24 UTC             98
10012017121415293700  2017-12-14 15:41:14 UTC             10
...                                       ...            ...
9982017103011564900   2017-10-30 12:05:27 UTC            129
9982017112108301300   2017-11-21 08:34:54 UTC            108
9982018012320471300   2018-01-23 20:52:34 UTC            144
9992017082616550000   2017-08-26 17:50:21 UTC            175
9992018010308503000   2018-01-03 09:16:00 UTC            186

variable                        end_station_name member_birth_year  \
trip_id
10002017102810135000                Bushrod Park            1983.0
10002017112118254500         2nd Ave at E 18th St            1968.0
10002018020418322600       Grand Ave at Perkins St            1962.0
10012017112918483500        Valencia St at 16th St            1983.0
10012017121415293700  Washington St at Kearny St               NaN
...                                          ...               ...
9982017103011564900         Harrison St at 20th St            1980.0
9982017112108301300          16th St Mission BART            1988.0
9982018012320471300                Precita Park            1991.0
9992017082616550000      49th St at Telegraph Ave            1990.0
9992018010308503000         Lakeside Dr at 14th St            1994.0

variable             member_gender               start_date start_station_id  \
trip_id
10002017102810135000          Male  2017-10-28 10:13:50 UTC              241
10002017112118254500        Female  2017-11-21 18:25:45 UTC              163
10002018020418322600          Male  2018-02-04 18:32:26 UTC              197
10012017112918483500          Male  2017-11-29 18:48:35 UTC              139
10012017121415293700           NaN  2017-12-14 15:29:37 UTC               44
...                            ...                      ...              ...
9982017103011564900           Male  2017-10-30 11:56:49 UTC              115
9982017112108301300           Male  2017-11-21 08:30:13 UTC              129
9982018012320471300           Male  2018-01-23 20:47:13 UTC              134
9992017082616550000           Male  2017-08-26 16:55:00 UTC              195
9992018010308503000           Male  2018-01-03 08:50:30 UTC              186

variable                                             start_station_name  \
trip_id
10002017102810135000                                 Ashby BART Station
10002017112118254500                          Lake Merritt BART Station
10002018020418322600                        El Embarcadero at Grand Ave
10012017112918483500              Garfield Square (25th St at Harrison St)
10012017121415293700  Civic Center/UN Plaza BART Station (Market St ...
...                                                                 ...
9982017103011564900                                  Jackson Playground
9982017112108301300                              Harrison St at 20th St
9982018012320471300                              Valencia St at 24th St
9992017082616550000                                  Bay Pl at Vernon St
9992018010308503000                              Lakeside Dr at 14th St

variable             subscriber_type zip_code
trip_id
10002017102810135000      Subscriber      NaN
10002017112118254500      Subscriber      NaN
10002018020418322600      Subscriber      NaN
10012017112918483500        Customer      NaN
10012017121415293700      Subscriber      NaN
...                              ...      ...
9982017103011564900       Subscriber      NaN
9982017112108301300       Subscriber      NaN
9982018012320471300       Subscriber      NaN
9992017082616550000         Customer      NaN
9992018010308503000       Subscriber      NaN

[10000 rows x 13 columns]
```

In [101…
```python
#view the new dataframe with transformation
df1.head(2)
df1.shape
```

Out[101…
```
(10000, 14)
```

In [12]:
```python
#Extract selected columns into new dataframe
df2 = df1[['start_date', 'end_date']].copy()
```

In [13]:
```python
df2.head(2)
```

Out[13]:

variable	start_date	end_date
trip_id		
10002017102810135000	2017-10-28 10:13:50 UTC	2017-10-28 10:27:54 UTC
10002017112118254500	2017-11-21 18:25:45 UTC	2017-11-21 18:33:46 UTC

In [14]:
```python
#view datatypes for new df
df2.dtypes
```

Out[14]:
```
variable
start_date    object
end_date      object
dtype: object
```

In [15]:
```python
#split the start date column
df3 = df2.start_date.str.split(expand=True)
df3.head(1)
```

Out[15]:

	0	1	2
trip_id			
10002017102810135000	2017-10-28	10:13:50	UTC

In [16]:
```python
#split the end date column
df4 = df2.end_date.str.split(expand=True)
df4.head(1)
```

Out[16]:

	0	1	2
trip_id			
10002017102810135000	2017-10-28	10:27:54	UTC

In [17]:
```python
#merge the two dataframes with split columns into one dataframe
df5 = pd.merge(df3, df4, left_index=True, right_index = True)
```

In [18]:
```python
#confirmation
df5.head(2)
```

Out[18]:

	0_x	1_x	2_x	0_y	1_y	2_y
trip_id						
10002017102810135000	2017-10-28	10:13:50	UTC	2017-10-28	10:27:54	UTC
10002017112118254500	2017-11-21	18:25:45	UTC	2017-11-21	18:33:46	UTC

In [19]:
```python
from datetime import datetime
```

In [20]:
```python
import time
```

In [21]:
```python
#convert timestamp to seconds
df5['1_x'] = pd.to_timedelta(df5['1_x'])
df5['1_y'] = pd.to_timedelta(df5['1_y'])
```

In [22]:
```python
df5['SDAsSeconds'] = df5['1_x'].dt.total_seconds()
```

In [23]:
```python
df5.head(2)
```

Out[23]:

	0_x	1_x	2_x	0_y	1_y	2_y	SDAsSeconds
trip_id							
10002017102810135000	2017-10-28	10:13:50	UTC	2017-10-28	10:27:54	UTC	36830.0
10002017112118254500	2017-11-21	18:25:45	UTC	2017-11-21	18:33:46	UTC	66345.0

In [24]:
```python
#convert
df5['EDAsSeconds'] = df5['1_y'].dt.total_seconds()
```

In [25]:
```python
df5.head(2)
```

Out[25]:

	0_x	1_x	2_x	0_y	1_y	2_y	SDAsSeconds	EDAsSeconds
trip_id								
10002017102810135000	2017-10-28	10:13:50	UTC	2017-10-28	10:27:54	UTC	36830.0	37674.0
10002017112118254500	2017-11-21	18:25:45	UTC	2017-11-21	18:33:46	UTC	66345.0	66826.0

In [26]:
```python
#find the difference in seconds between ED and SD
difference = (df5['EDAsSeconds'] - df5['SDAsSeconds'])
```

In [27]:
```python
#conver the difference into minutes
difference1 = (difference / 60)
difference1
```

Out[27]:
```
trip_id
10002017102810135000    14.066667
10002017112118254500     8.016667
10002018020418322600     3.050000
10012017112918483500     9.816667
10012017121415293700    11.616667
                          ...
9982017103011564900      8.633333
9982017112108301300      4.683333
9982018012320471300      5.350000
9992017082616550000     55.350000
9992018010308503000     25.500000
Length: 10000, dtype: float64
```

In [28]:
```python
#add new column into dataframe with difference in minutes
df5['time'] = difference1
```

In [29]:
```python
#sort df by time
sorted_df = df5.sort_values(by='time')
```

In [30]:
```python
#create customer segments based on time value
#if time is greater than 2 or time is equal or less than 2
segment = sorted_df.time.apply(lambda x: 1 if x>2 else 0)
```

In [31]:
```python
sorted_df['segment'] = segment
sorted_df.tail(1)
```

Out[31]:

	0_x	1_x	2_x	0_y	1_y	2_y	SDAsSeconds	EDAsSeconds	time	segment
trip_id										
20620131121010500	2013-11-21	01:05:00	UTC	2013-11-21	17:42:00	UTC	3900.0	63720.0	997.0	1

In [32]:
```python
sorted_df.dtypes
```

Out[32]:
```
0_x                      object
1_x             timedelta64[ns]
2_x                      object
0_y                      object
1_y             timedelta64[ns]
2_y                      object
SDAsSeconds             float64
EDAsSeconds             float64
time                    float64
segment                   int64
dtype: object
```

In [33]:
```python
#find the unique values of
sorted_df.segment.unique()
```

Out[33]:
```
array([0, 1], dtype=int64)
```

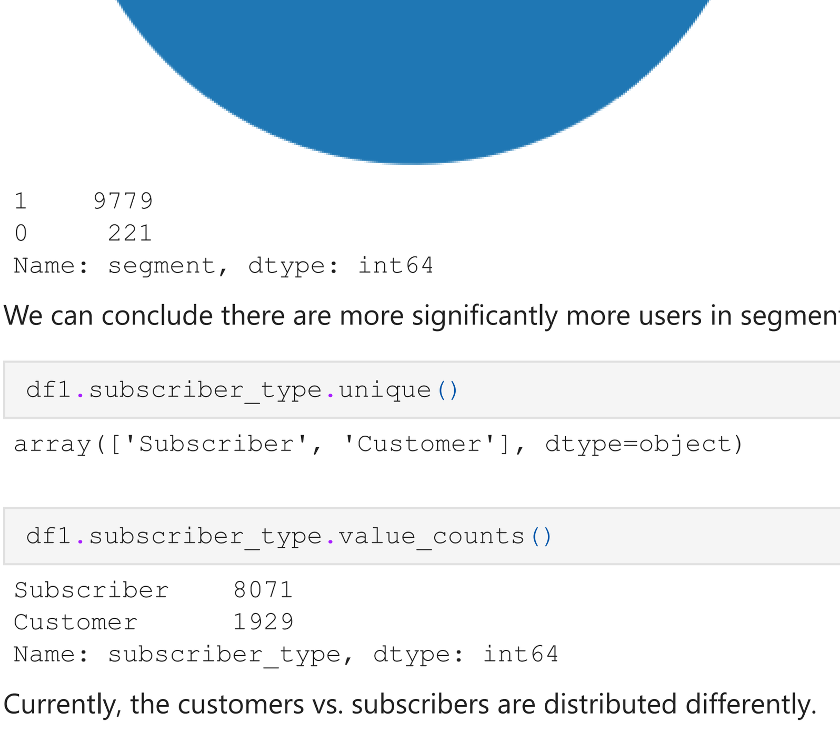
The users are segemented into two different groups based on time spent using the bike. If the time spent is greater than 2, users are put in one segment where subscription should be advertised. If time spent is less than or equal to 2, single rides should be advertised

In [42]:
```python
#visualize segments based on time
import matplotlib.pyplot as plt
```

In [35]:
```python
values = sorted_df['segment'].value_counts()
```

In [37]:
```python
labels = sorted_df['segment'].unique().tolist()
```

In [41]:
```python
plt.pie(values, labels = labels, radius =2)
plt.show()
print(values)
```




```
1    9779
0     221
Name: segment, dtype: int64
```

We can conclude there are more significantly more users in segment 1 than segment 0

In [43]:
```python
df1.subscriber_type.unique()
```

Out[43]:
```
array(['Subscriber', 'Customer'], dtype=object)
```

In [49]:
```python
df1.subscriber_type.value_counts()
```

Out[49]:
```
Subscriber    8071
Customer      1929
Name: subscriber_type, dtype: int64
```

Currently, the customers vs. subscribers are distributed differently.