```
In [1]: # task 3
        # import libraries for this task
        import pandas as pd
        import json
        import datetime
```

Step 1: Network overlap: with the "following.json" dataset, write a function that takes any two influencers' user ID's and calculate the fraction of followers these two influencers share over the total number of followers of the less followed influencer. The reference date is April 30, 2022.

```
In [76]: # import following.json into the same directory (on anacondas jupyter notebook) and import it into this file
         # for analysis.
         following_df = pd.read_json('following.json')
```

```
In [77]: following_df
```

Out[77]:

	follower_uid	influencer_uid	follow_timestamp
0	829451382935531520	902200087	2022-04-26 16:07:33 UTC
1	2872039840	969221141347913728	2022-02-18 15:51:26 UTC
2	1499994241979801600	14709326	2022-04-23 05:40:07 UTC
3	1135376232839864320	836533158669500416	2022-04-28 13:26:38 UTC
4	431066540	16106584	2017-04-14 11:35:28 UTC
...	...	...	...
551130	78333251	158414847	2013-11-11 01:20:59 UTC
551131	2802988291	158414847	2020-01-11 14:50:13 UTC
551132	1519077442006261760	158414847	2022-04-27 01:35:00 UTC
551133	2861786021	158414847	2019-07-21 14:38:03 UTC
551134	1518751896810496000	158414847	2022-04-26 00:48:35 UTC

551135 rows × 3 columns

```
In [4]: following_df.head() # look at the structure of the dataframe to see how i can format the function
```

Out[4]:

	follower_uid	influencer_uid	follow_timestamp
0	829451382935531520	902200087	2022-04-26 16:07:33 UTC
1	2872039840	969221141347913728	2022-02-18 15:51:26 UTC
2	1499994241979801600	14709326	2022-04-23 05:40:07 UTC
3	1135376232839864320	836533158669500416	2022-04-28 13:26:38 UTC
4	431066540	16106584	2017-04-14 11:35:28 UTC

```
In [11]: following_df.dtypes # check the data types - we can see the 'follow_timestamp' column is a of type str.

Out[11]: follower_uid         int64
         influencer_uid       int64
         follow_timestamp    object
         dtype: object
```

```
In [78]: # function for step 1)

         # the set up of this function assumes that any dataframe u pass in has the same column names
         # as the "following_df" dataset.

         def shared_followers(df: pd.DataFrame, influencer1_id, influencer2_id):

             # filter data to include only entries from April 30, 2022
             filtered_data = df[df['follow_timestamp'].str.startswith('2022-04-30')]

             # get set of follower IDs for each influencer (each follower is unique)
             influencer1_followers = set(filtered_data[filtered_data['influencer_uid'] == influencer1_id]['follower_uid'])
             influencer2_followers = set(filtered_data[filtered_data['influencer_uid'] == influencer2_id]['follower_uid'])

             # calculate total number of unique followers for each influencer that they both share
             overlap_count = len(influencer1_followers.intersection(influencer2_followers))

             # calculate total number of followers of the less followed influencer
             total_followers = min(len(influencer1_followers), len(influencer2_followers))

             # calculate fraction of followers that two influencers share
             if total_followers > 0: # just in case the total_followers of that influencer are 0, we don't want a
                 # ZeroDivError.

                 overlap_fraction = overlap_count / total_followers
             else:
                 return 0.0

                 # just made the function return 0.0 when the lower influencer has 0 followers (no intersection between
                 # any followers in this case)

             return overlap_fraction
```

```
In [79]: # example usage:
         shared = shared_followers(following_df, 158414847, 158414847)
         print(shared)
         # makes sense that the same influencer has a fraction of 1.0 (all followers are same) on april 30th.

         1.0
```

Step 2: Engagement overlap: with the "engagement.json" dataset, write a function that takes any two influencers' user ID's and calculate the fraction of engagers of these two influencers' tweets as a function of the total number of engagers of the less engaged influencer, over the period of April 22, 2022 to April 30, 2022.

```
In [54]: # import engagement.json into the same directory (on anacondas jupyter notebook) and import it into this file
         engagement_df = pd.read_json('engagement.json')
```

```
In [55]: engagement_df.head()
```

Out[55]:

	follower_uid	influencer_uid	engaged_tweetID	engaged_dt
0	1000041135396433920	1298372735383605248	1520185102143303680	2022-04-29
1	1000041135396433920	158414847	1520185102143303680	2022-04-29
2	1000041135396433920	288277167	1520185102143303680	2022-04-29
3	1000146462112665600	1022693675250249728	1518276445335961600	2022-04-24
4	1000146462112665600	10774652	1518276445335961600	2022-04-24

```
In [82]: engagement_df.dtypes # engaged_dt is an object

Out[82]: follower_uid        int64
         influencer_uid      int64
         engaged_tweetID     int64
         engaged_dt         object
         dtype: object
```

```
In [101… def shared_engagements(dataset, influencer1_id, influencer2_id):
             # Step 1: Filter dataset based on date range and influencer IDs
             filtered_data = dataset[(dataset['engaged_dt'] >= '2022-04-22') & (dataset['engaged_dt'] <= '2022-04-30') &
                                     ((dataset['influencer_uid'] == influencer1_id) | (dataset['influencer_uid'] == influencer2_id))]

             # Step 2: Group by influencer ID, engaged tweet ID, and follower ID, count unique tweet IDs
             grouped_data = filtered_data.groupby(['influencer_uid', 'engaged_tweetID', 'follower_uid']).size().reset_index(name='count')

             # Step 3: Get the total number of unique followers who engaged for each influencer
             influencer1_engagement = grouped_data[grouped_data['influencer_uid'] == influencer1_id]['follower_uid'].nunique()
             influencer2_engagement = grouped_data[grouped_data['influencer_uid'] == influencer2_id]['follower_uid'].nunique()

             # Step 4: Determine the less engaged influencer
             if influencer1_engagement < influencer2_engagement:
                 less_engaged_followers = influencer1_engagement
             else:
                 less_engaged_followers = influencer2_engagement

             # Step 5: Get unique followers who engaged in both influencers' tweets
             shared_engagers = set(grouped_data[grouped_data['influencer_uid'] == influencer1_id]['follower_uid']).intersection(
                 set(grouped_data[grouped_data['influencer_uid'] == influencer2_id]['follower_uid'])
             )

             # Step 6: Calculate the fraction of shared engagers over the total unique followers of the less engaged influencer
             if less_engaged_followers > 0:
                 fraction_engagers = len(shared_engagers) / less_engaged_followers
             else:
                 return 0.0

             return fraction_engagers
```

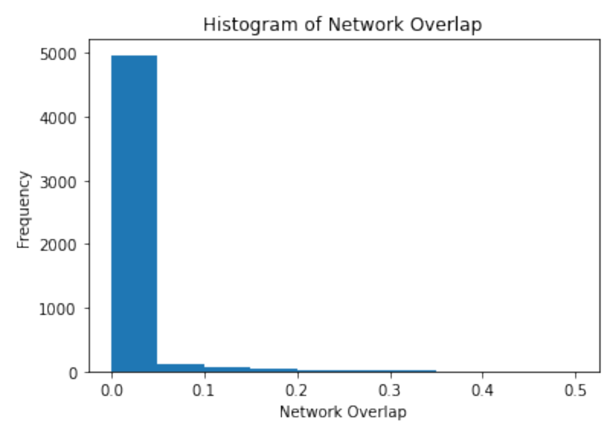
```
In [102… shared = shared_engagements(engagement_df, 158414847, 158414847)
         print(shared)
         # 2 same influencers have a 1.0 ratio of shared engagement of tweets within the same period of time.

         1.0
```

Step 3: Produce two histograms of network overlap (Step 2) and engagement overlap (Step 3) measures, respectively, across all influencer pairs

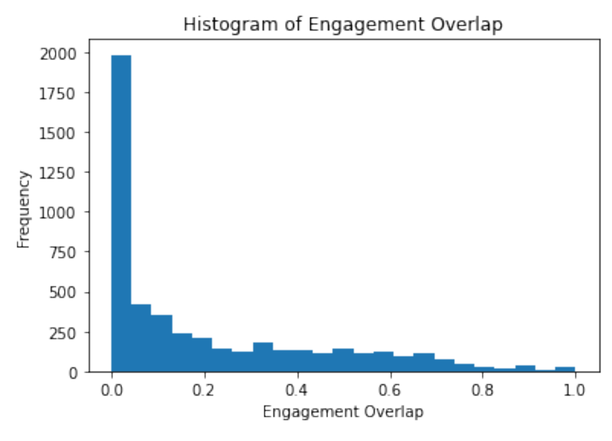
```
In [70]: import matplotlib.pyplot as plt
```

```
In [ ]: # network overlap - fraction of same followers on april 30th, bw 2 influencers
        # engagement overlap - fraction of engaged tweets bw 22-30 of two influencers
```

```
In [80]: # network overlap histogram

         network_overlap = []
         influencer_ids = list(following_df['influencer_uid'].unique())

         # loop through all possible influencer pairs
         for i in range(len(influencer_ids)):
             for j in range(i+1, len(influencer_ids)):
                 # call shared_followers functions
                 network_overlap.append(shared_followers(following_df, influencer_ids[i], influencer_ids[j]))

         # create histogram for network overlap
         plt.hist(network_overlap)
         plt.xlabel('Network Overlap')
         plt.ylabel('Frequency')
         plt.title('Histogram of Network Overlap')
         plt.show()
```




```
In [103… # engagement overlap histogram
         engagement_overlap = []
         influencer_ids_2 = list(engagement_df['influencer_uid'].unique())

         for i in range(len(influencer_ids_2)):
             for j in range(i+1, len(influencer_ids_2)):
                 # call shared_engagements functions
                 engagement_overlap.append(shared_engagements(engagement_df, influencer_ids_2[i], influencer_ids_2[j]))

         # create histogram for engagement overlap
         plt.hist(engagement_overlap, bins=23)
         plt.xlabel('Engagement Overlap')
         plt.ylabel('Frequency')
         plt.title('Histogram of Engagement Overlap')
         plt.show()
```



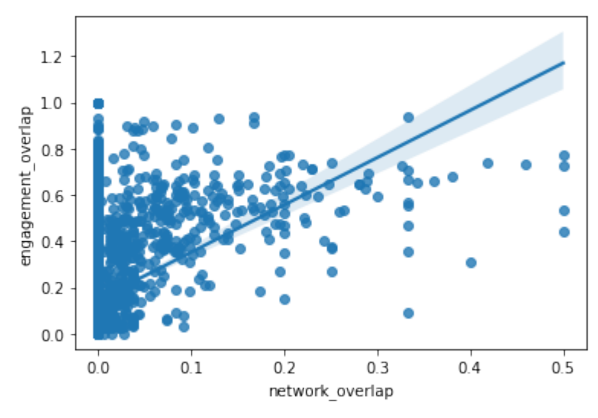
Step 5: Use OLS to regress engagement overlap on network overlap measures for all influencers pairs, and plot the regression results on a two-dimensional graph with standard error bands.

```
In [ ]: # since the two influencer pairs have different lengths overthe two dataframes, im going to only use
        # the pairs that are present in both. create an intersection between the lists first and then find the
        # overlaps for those to run the regression, since we can't run regressions on 2 lists of different lengths.
        new_set = list(set(influencer_ids) & set(influencer_ids_2))
        new_set
```

```
In [82]: # then find the network and engagement overlaps for only these influencers.
         # not going to print the histograms for this: just the list
         new_network_overlap = []

         for i in range(len(new_set)):
             for j in range(i+1, len(new_set)):
                 # call shared_followers functions
                 new_network_overlap.append(shared_followers(following_df, influencer_ids[i], influencer_ids[j]))
```

```
In [105… new_engagement_overlap = []

         for i in range(len(new_set)):
             for j in range(i+1, len(new_set)):
                 # call shared_engagements functions
                 new_engagement_overlap.append(shared_engagements(engagement_df, influencer_ids[i], influencer_ids[j]))
```

```
In [106… import statsmodels.api as sm
         import seaborn as sns

         # create a dataframe with network overlap and engagement overlap measures
         df_overlap = pd.DataFrame({'network_overlap': new_network_overlap,
                                    'engagement_overlap':new_engagement_overlap })

         # perform OLS regression
         X = sm.add_constant(df_overlap['network_overlap'])
         model = sm.OLS(df_overlap['engagement_overlap'], X)
         results = model.fit()


         # plot the results with standard error bands
         sns.regplot(x='network_overlap', y='engagement_overlap', data=df_overlap, ci=95) #95th confidence interval

Out[106]: <AxesSubplot:xlabel='network_overlap', ylabel='engagement_overlap'>
```



Step 6: Develop a hypothesis on the determinants of the difference between network vs engagement overlaps, i.e. what makes two influencers have high network overlap but low engagement overlap and vice versa?

One possible hypothesis on the determinants of the difference between network vs engagement overlaps could be related to the content and style of the tweets that the influencers are posting. For instance, two influencers might have a high network overlap because they have similar interests and followers, but their engagement overlap might be low if their tweeting styles are different or if they are covering different topics. Similarly, two influencers might have a low network overlap but a high engagement overlap if they are posting about similar topics but have different audiences. Another factor that could impact the difference between network vs engagement overlaps could be the frequency and timing of the tweets, as influencers might have different tweeting schedules and patterns that could affect how their followers engage with their content. Overall, there could be multiple factors that contribute to the difference between network vs engagement overlaps, and further research and analysis would be needed to identify and test these hypotheses.