Appendix

1.a.

In [14]:
```
# Determine the number of locations.
num_loc = len(pd.unique(nc['sub_icb_location_name']))


print(" The number of unqiue locations is:",
      (num_loc))
```

```
 The number of unqiue locations is: 106
```

In [17]:
```
# Determine the number of service settings.
num_service_setting = len(pd.unique(nc['service_setting']))

print("Number of service settings is:", num_service_setting)
```

```
Number of service settings is: 5
```

In [18]:
```
# Determine the number of context types.
num_context_type = len(pd.unique(nc['context_type']))

print(" The number of unqiue locations is:",
      (num_context_type))
```

```
 The number of unqiue locations is: 3
```

In [19]:
```
# Determine the number of national categories.
num_nat_cat = len(pd.unique(nc['national_category']))

print("Number of national categories is:", num_nat_cat)
```

```
Number of national categories is: 18
```

In [20]:
```
# Determine the number of appointment status.
num_app_status = len(pd.unique(ar['appointment_status']))

print("Number of appointment statuses is:", num_app_status)
ar.appointment_status.value_counts()
```

```
Number of appointment statuses is: 3
```

Out[20]:
```
Attended    232137
Unknown     201324
DNA         163360
Name: appointment_status, dtype: int64
```

2.a.

In [309]:
```
# Check the top five locations based on record count from nc
nc['sub_icb_location_name'].value_counts().head()
```

Out[309]:
```
NHS North West London ICB - W2U3Z              13007
NHS Kent and Medway ICB - 91Q                  12637
NHS Devon ICB - 15N                            12526
NHS Hampshire and Isle Of Wight ICB - D9Y0V    12171
NHS North East London ICB - A3A8R              11837
Name: sub_icb_location_name, dtype: int64
```

In [1053]:
```
# Create a DataFrame to log the location with the most records
nth_west_ldn = nc[nc['sub_icb_location_name']
                  == 'NHS North West London ICB - W2U3Z']
# View the DataFrame
nth_west_ldn.head()
```

In [337]:
```
# Determine the sum of appointments per each service setting
nth_west_ldn.groupby(['service_setting'])\
    .sum('count_of_appointments').reset_index().sort_values(by='count_of_appointments', ascending=False)
```

Out[337]:

	service_setting	count_of_appointments
1	General Practice	10432225
4	Unmapped	904234
2	Other	343642
3	Primary Care Network	240283
0	Extended Access Provision	222006

In [955]:
```
# Determine the count of records for each service setting
nth_west_ldn['service_setting'].value_counts()
```

Out[955]:
```
General Practice             4609
Other                        2858
Primary Care Network         2791
Extended Access Provision    2415
Unmapped                      334
Name: service_setting, dtype: int64
```

2.b.

In [380]:
```
# Determine the bottom location based on record count
nc['sub_icb_location_name'].value_counts().tail()
```

Out[380]:
```
NHS North East and North Cumbria ICB - 00N    4210
NHS Lancashire and South Cumbria ICB - 02G    4169
NHS Cheshire and Merseyside ICB - 01V         3496
NHS Cheshire and Merseyside ICB - 01T         3242
NHS Greater Manchester ICB - 00V              2170
Name: sub_icb_location_name, dtype: int64
```

In [382]:
```
# Create a DataFrame to log the location with the most records
grt_manc_00v = nc[nc['sub_icb_location_name']
                  == 'NHS Greater Manchester ICB - 00V']
# View the DataFrame
grt_manc_00v
```

In [383]:
```
# Determine the sum of appointments per each service setting
grt_manc_00v.groupby(['service_setting'])\
    .sum('count_of_appointments').reset_index().sort_values(by='count_of_appointments', ascending=False)
```

Out[383]:

	service_setting	count_of_appointments
4	Unmapped	430394
1	General Practice	207797
0	Extended Access Provision	942
3	Primary Care Network	74
2	Other	4

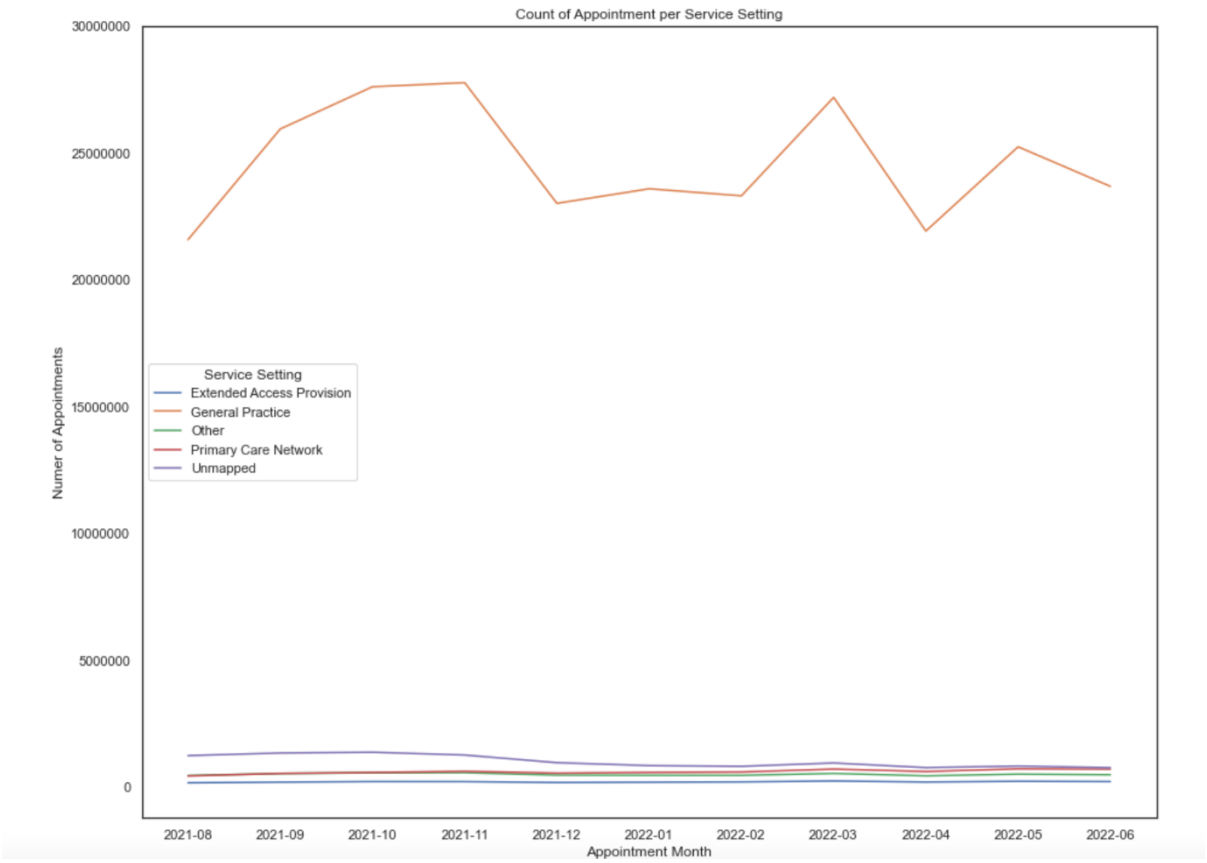
In [384]:
```
# Determine the record count of each service setting
grt_manc_00v['service_setting'].value_counts()
```

Out[384]:
```
General Practice             1586
Extended Access Provision     281
Unmapped                      279
Primary Care Network           22
Other                           2
Name: service_setting, dtype: int64
```

3.a.

In [525]:
```
# Aggregate on monthly level and determine the sum of records per month.
nc_ss = nc.groupby(['appointment_month', 'service_setting']
                   ).sum().reset_index()
# View output.
nc_ss.head()
```

In [528]:
```
# Plot the appointments over the available date range, and review the service settings for months.
# Create a lineplot.
nc_ss_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',
                        hue='service_setting', data=nc_ss, ci=None)

# Customise and tidy the plot.
nc_ss_ax.legend(title='Service Setting', loc='center left')
nc_ss_ax.set_xlabel('Appointment Month')
nc_ss_ax.set_ylabel('Numer of Appointments')
nc_ss_ax.set_title("Count of Appointment per Service Setting")
nc_ss_ax.set_yticks([0, 5000000, 10000000, 15000000,
                     20000000, 25000000, 30000000])
nc_ss_ax.set_yticks([2500000, 7500000, 12500000, 17500000,
                     22500000, 27500000], minor=True)
nc_ss_ax.set_yticklabels(
    ['0', '5000000', '10000000', '15000000', '20000000', '25000000', '30000000'])
```



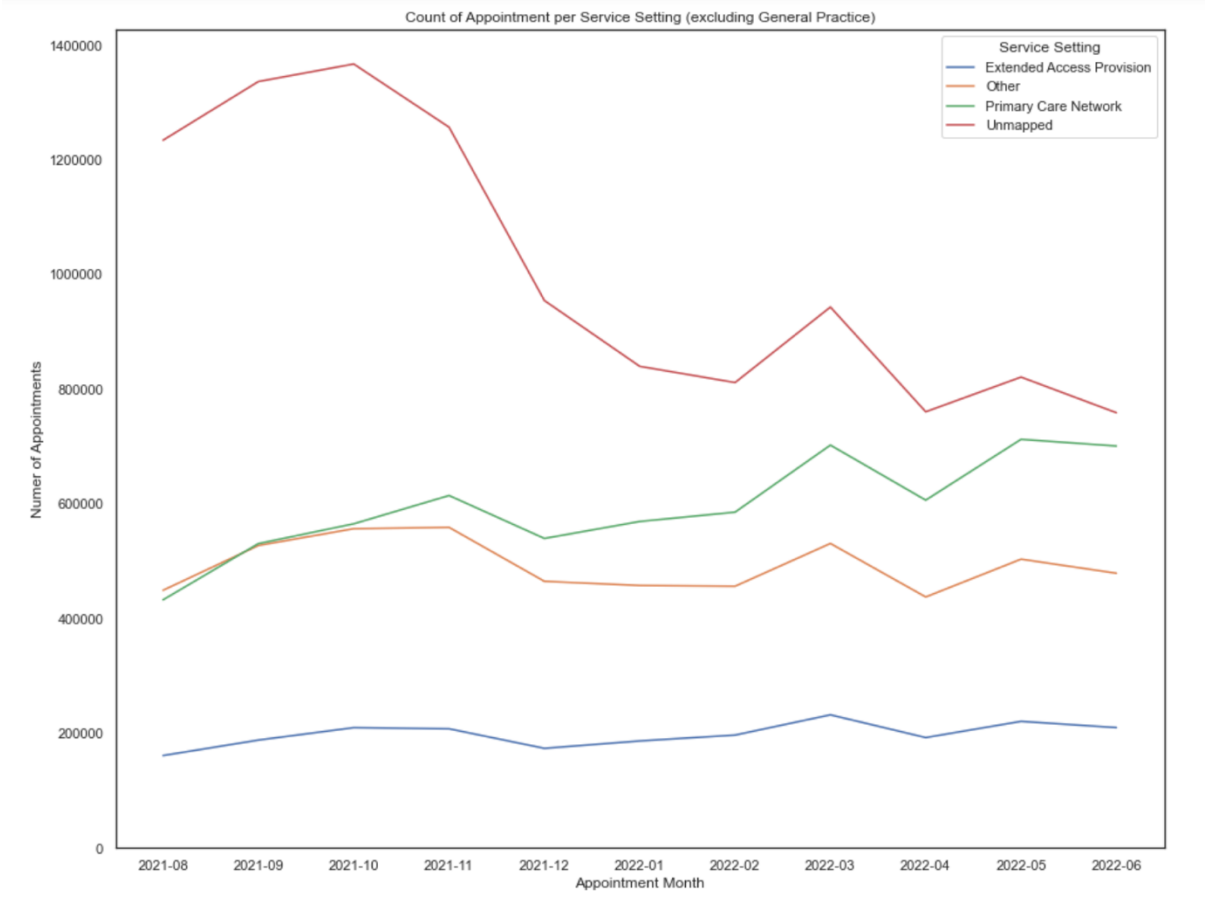
3.b.

In [434]:
```
# Subset the data to remove General Practice
nc_ex_gp = nc_ss[nc_ss['service_setting'] != 'General Practice']
# View DataFrame
nc_ex_gp.head()
```

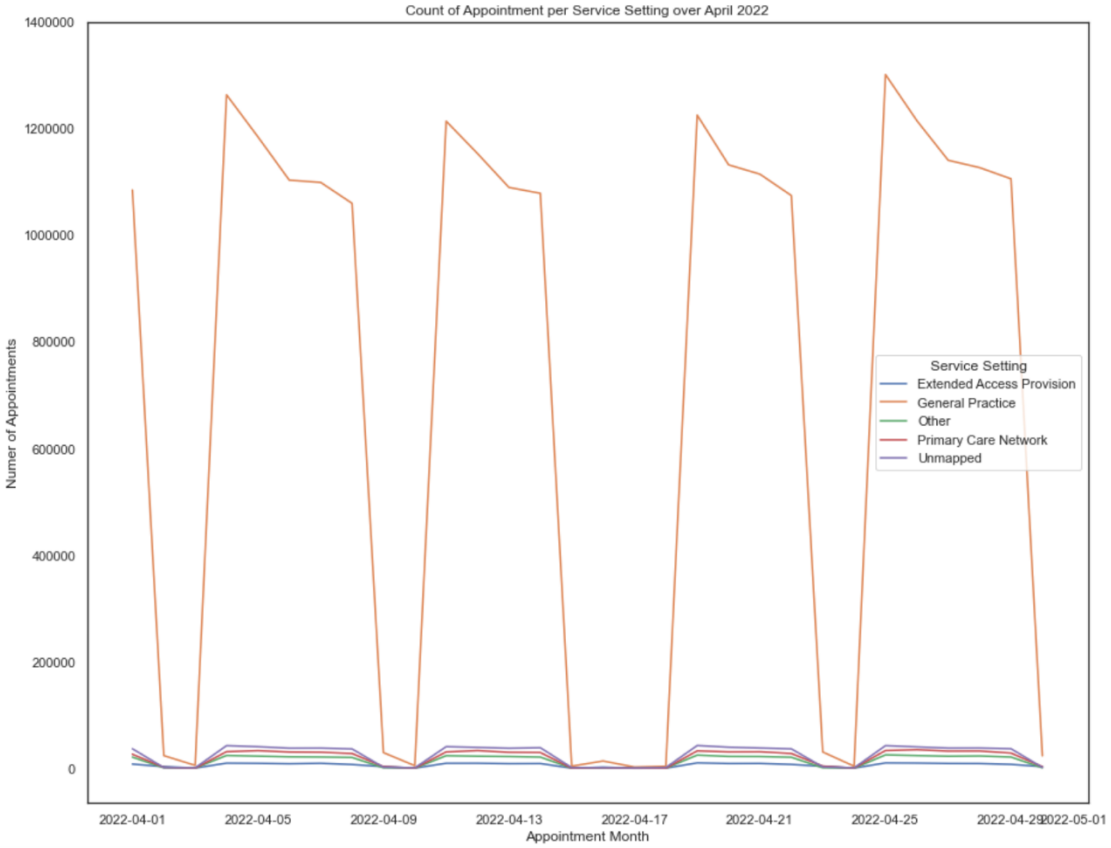
Out[434]:

	appointment_month	service_setting	count_of_appointments
0	2021-08	Extended Access Provision	160927
2	2021-08	Other	449101
3	2021-08	Primary Care Network	432448
4	2021-08	Unmapped	1233843
5	2021-09	Extended Access Provision	187906

In [530]:
```
# Create a lineplot showing service setting excluding General Practice
ex_gp_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',
                        hue='service_setting', data=nc_ex_gp, ci=None)

# Customise and tidy the plot.
ex_gp_ax.set_xlabel('Appointment Month')
ex_gp_ax.set_ylabel('Numer of Appointments')
ex_gp_ax.set_title(
    "Count of Appointment per Service Setting (excluding General Practice)")
ex_gp_ax.legend(title='Service Setting', loc='upper right')
ex_gp_ax.set_yticks([0, 200000, 400000, 600000,
                     800000, 1000000, 1200000, 1400000])
ex_gp_ax.set_yticks([100000, 300000, 500000, 700000,
                      900000, 1100000, 1300000], minor=True)
ex_gp_ax.set_yticklabels(
    ['0', '200000', '400000', '600000',
                     '800000', '1000000', '1200000', '1400000'])
```



3.c.



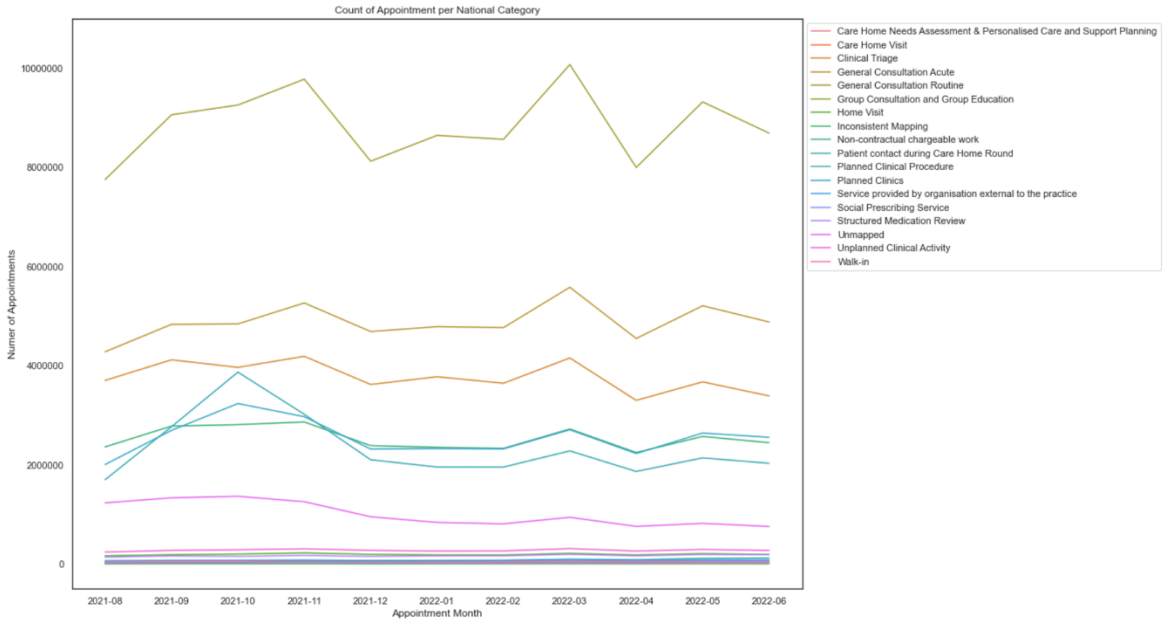
4.a.

In [68]:
```
# Create a separate data set that can be used in future weeks.
nc_nc = nc.groupby(
    ['appointment_month', 'national_category']).sum().reset_index()
# View output.
nc_nc.head()
```

Out[68]:

	appointment_month	national_category	count_of_appointments
0	2021-08	Care Home Needs Assessment & Personalised Care...	29676
1	2021-08	Care Home Visit	47583
2	2021-08	Clinical Triage	3704207
3	2021-08	General Consultation Acute	4280920
4	2021-08	General Consultation Routine	7756045

In [581]:
```
# Plot the appointments over the available date range, and review the national categories for months.
# Create a lineplot.
nc_nc_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',
                        hue='national_category', data=nc_nc, ci=None)

# Customise and tidy the plot
nc_nc_ax.legend(bbox_to_anchor=(1, 1))
nc_nc_ax.set_xlabel('Appointment Month')
nc_nc_ax.set_ylabel('Numer of Appointments')
nc_nc_ax.set_title("Count of Appointment per National Category")
nc_nc_ax.set_yticks([0, 2000000, 4000000, 6000000, 8000000, 10000000])
nc_nc_ax.set_yticks([1000000, 3000000, 5000000, 7000000,
                     9000000, 11000000], minor=True)
nc_nc_ax.set_yticklabels(['0', '2000000', '4000000', '6000000',
                          '8000000', '10000000'])
```



4.b.

In [549]:
```
# Create a new DataFrame featuring the national categories with 10 million + appointments
nc_nc_top = nc_nc[(nc_nc['national_category'].isin(['General Consultation Routine',
                                                    'General Consultation Acute',
                                                    'Clinical Triage', 'Planned Clinics',
                                                    'Inconsistent Mapping', 'Planned Clinical Procedure',
                                                    'Unmapped']))]

# View the DataFrame
nc_nc_top
```

Out[549]:

	appointment_month	national_category	count_of_appointments
2	2021-08	Clinical Triage	3704207
3	2021-08	General Consultation Acute	4280920
4	2021-08	General Consultation Routine	7756045
7	2021-08	Inconsistent Mapping	2363093
10	2021-08	Planned Clinical Procedure	1701814
...	...	...	...
184	2022-06	General Consultation Routine	8691384
187	2022-06	Inconsistent Mapping	2447611
190	2022-06	Planned Clinical Procedure	2032073
191	2022-06	Planned Clinics	2556175
195	2022-06	Unmapped	758640

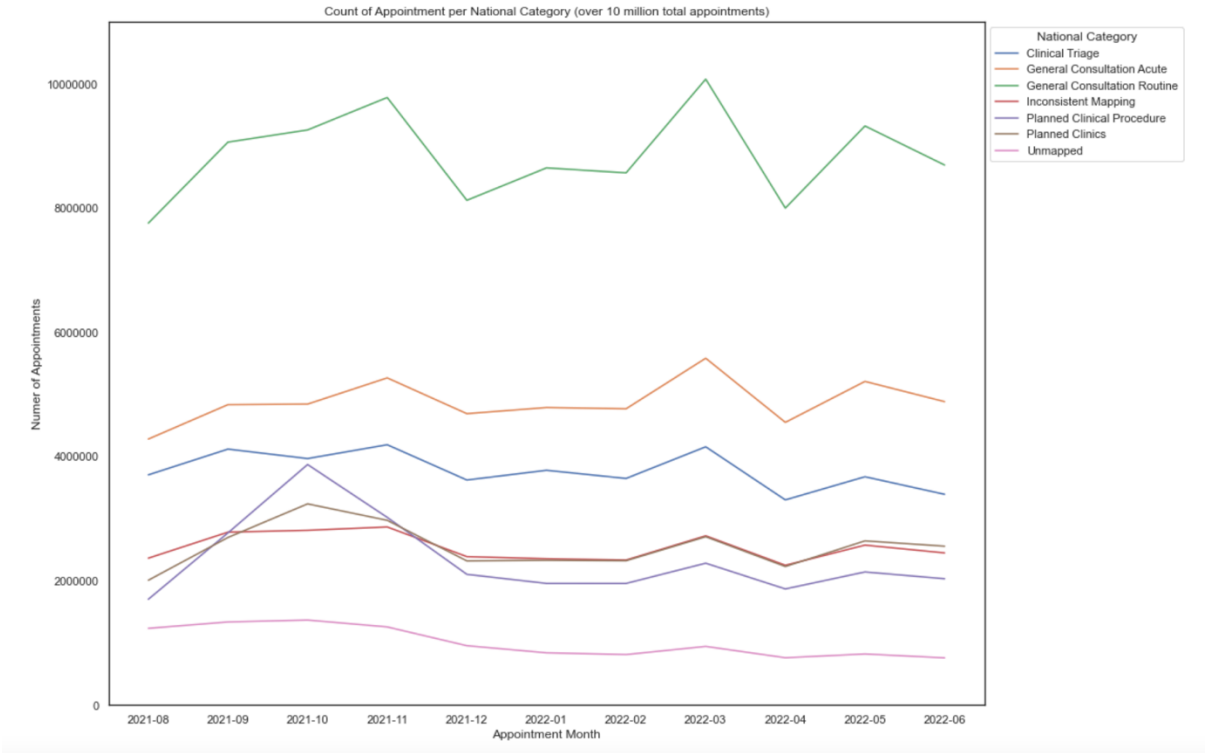
In [522]:
```
# Great a new DataFrame to include all national categories which have less than 10 million appointments
nc_nc_bottom = nc_nc[(nc_nc['national_category'].isin(['Unplanned Clinical Activity',
                                                       'Home Visit', 'Structured Medication Review',
                                                       'Service provided by organisation external to the practice',
                                                       'Patient contact during Care Home Round',
                                                       'Care Home Visit', 'Social Prescribing Service',
                                                       'Walk-in',
                                                       'Care Home Needs Assessment & Personalised Care and Support Plan
                                                       'Non-contractual chargeable work',
                                                       'Group Consultation and Group Education']))]
nc_nc_bottom
```

Out[522]:

	appointment_month	national_category	count_of_appointments
0	2021-08	Care Home Needs Assessment & Personalised Care and Support Planning	29676
1	2021-08	Care Home Visit	47583
5	2021-08	Group Consultation and Group Education	5161
6	2021-08	Home Visit	165061
8	2021-08	Non-contractual chargeable work	10775
...	...	...	...
192	2022-06	Service provided by organisation external to the practice	81663
193	2022-06	Social Prescribing Service	55066
194	2022-06	Structured Medication Review	187800
196	2022-06	Unplanned Clinical Activity	274491
197	2022-06	Walk-in	35935

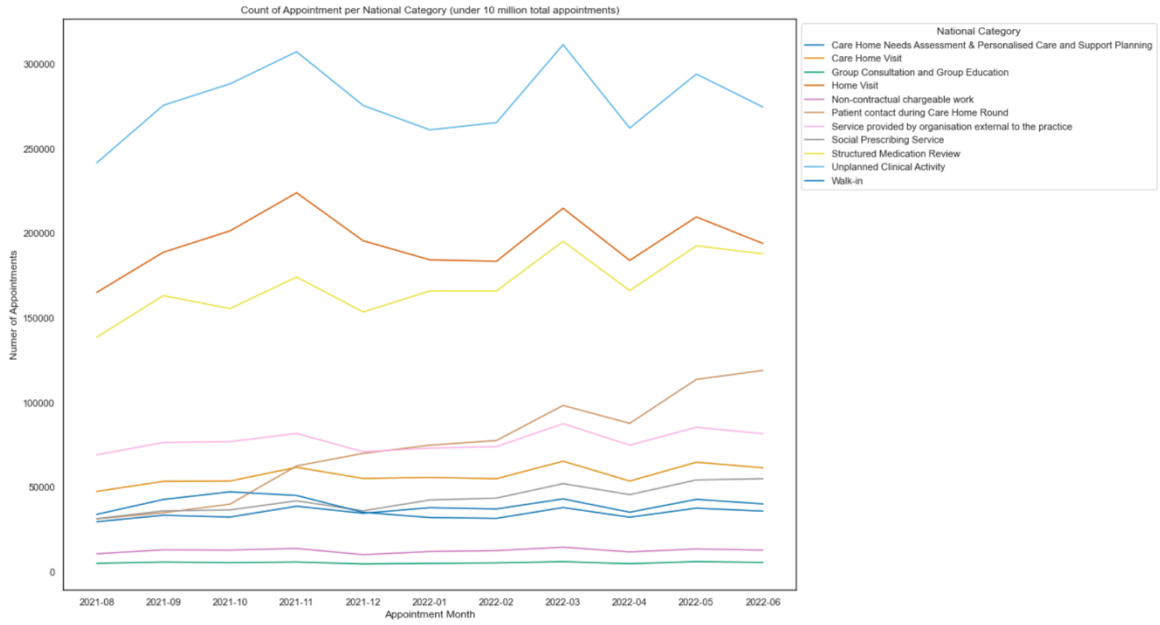
4.c.

In [609]:
```
# Plot the appointments over the available date range, and review the top national categories for months.
# Create a lineplot.
top_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',
                      hue='national_category', data=nc_nc_top, ci=None)
# Customise and tidy the plot
top_ax.legend(title='National Category', bbox_to_anchor=(1, 1))
top_ax.set_xlabel('Appointment Month')
top_ax.set_ylabel('Numer of Appointments')
top_ax.set_title("Count of Appointment per National Category (over 10 million total appointments)")
top_ax.set_yticks([0, 2000000, 4000000, 6000000, 8000000, 10000000])
top_ax.set_yticks([1000000, 3000000, 5000000, 7000000,
                   9000000, 11000000], minor=True)
top_ax.set_yticklabels(['0', '2000000', '4000000', '6000000',
                        '8000000', '10000000'])
```



4.d.

In [610]:
```
# Plot the appointments over the available date range, and review the bottom national categories for months.
# Create a lineplot.
bottom_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',
                         hue='national_category', palette='colorblind', data=nc_nc_bottom, ci=None)

# Customise and tidy the plot
bottom_ax.legend(title='National Category', bbox_to_anchor=(1, 1))
bottom_ax.set_xlabel('Appointment Month')
bottom_ax.set_ylabel('Numer of Appointments')
bottom_ax.set_title("Count of Appointment per National Category (under 10 million total appointments)")
```



5.a.

In [140]:
```
# Loop through the messages, and create a list of values containing the # symbol.
tags = []

for y in [x.split(' ') for x in tweets['tweet_full_text'].values]:
    for z in y:
        if '#' in z:
            # Change to lowercase.
            tags.append(z.lower())

tags
```

In [141]:
```
# Display the first 30 records.
hashtags = pd.Series(tags).value_counts()
hashtags.iloc[:30]
```

Out[141]:

#healthcare	716
#health	80
#medicine	41
#ai	40
#job	38
#medical	35
#strategy	30
#pharmaceutical	28
#digitalhealth	25
#pharma	25
#marketing	25
#medtwitter	24

5.b.

In [142]:
```
# Convert the series to a DataFrame in preparation for visualisation.
data = pd.DataFrame(hashtags).reset_index()
data.rename(columns={data.columns[0]: 'word',
            data.columns[1]: 'count'}, inplace=True)
# Rename the columns.
data
```

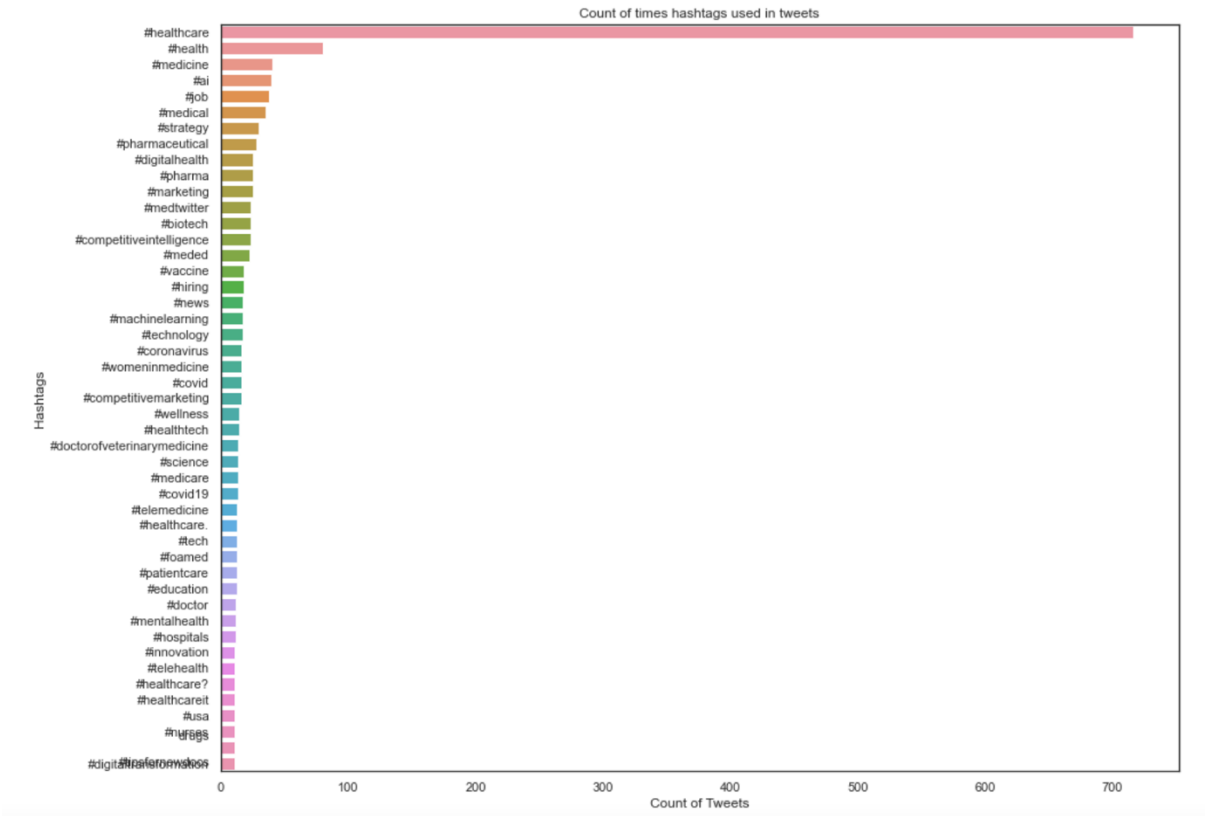
Out[142]:

	word	count
0	#healthcare	716
1	#health	80
2	#medicine	41
3	#ai	40
4	#job	38
...	...	...
1749	#evestudy	1
1750	#patientdata...	1
1751	#secure	1
1752	#sms	1
1753	\n#csjmu	1

1754 rows × 2 columns

5.c.

In [664]:
```
# Create a Seaborn barplot indicating records with a count >10 records.
hashtag_ax = sns.barplot(
    x='count', y='word', estimator=sum, data=data_above_ten)

# Customise and tidy the plot
hashtag_ax.set_xlabel('Count of Tweets')
hashtag_ax.set_ylabel('Hashtags')
hashtag_ax.set_title(
    "Count of times hashtags used in tweets")
```



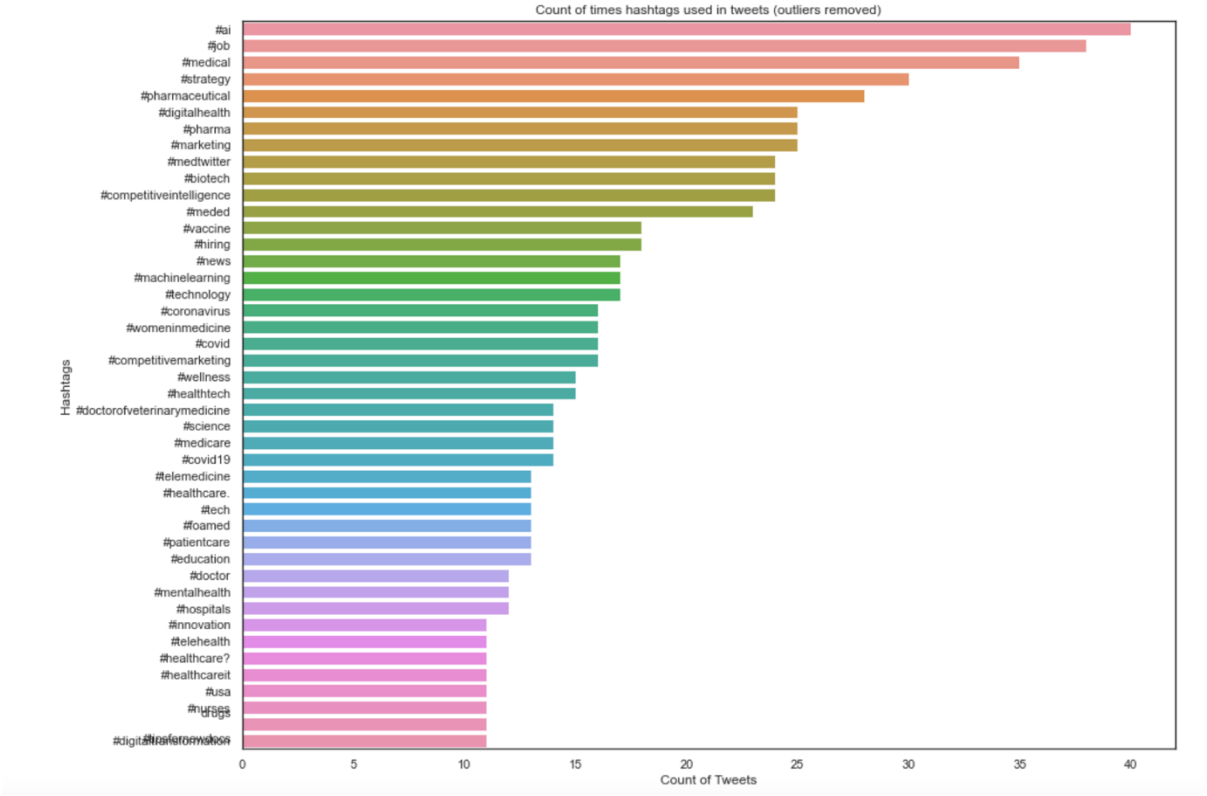
5.d.

In [148]:
```
# The columns you want to search for outliers in.
cols = ['count']

# Calculate quantiles and IQR.
# Same as np.percentile but maps (0,1) and not (0,100).
Q1 = data_above_ten[cols].quantile(0.25)
Q3 = data_above_ten[cols].quantile(0.75)
IQR = Q3 - Q1
IQR

# Return a Boolean array of the rows with (any) non-outlier column values.
condition = ~((data_above_ten[cols] < (Q1 - 1.5 * IQR))
              | (data_above_ten[cols] > (Q3 + 1.5 * IQR))).any(axis=1)

# Filter our DataFrame based on condition.
data_above_ten_non_outlier = data_above_ten[condition]
data_above_ten_non_outlier.shape
```

Out[148]: (44, 2)

5.e.

In [665]:
```
# Create the plot.
# View the barplot.
non_outlier_ax = sns.barplot(x='count', y='word', estimator=sum,
                             data=data_above_ten_non_outlier)

# Customise and tidy the plot
non_outlier_ax.set_xlabel('Count of Tweets')
non_outlier_ax.set_ylabel('Hashtags')
non_outlier_ax.set_title(
    "Count of times hashtags used in tweets (outliers removed)")
```



6.a.

In [1017]:
```
# Filter the data set to only look at data from 2021-08 onwards.
ar_2 = ar[ar['appointment_month'] >= '2021-08']
ar_2
```

Out[1017]:

	icb_ons_code	appointment_month	appointment_status	hcp_type	appointment_mode	time_between_book_and_appointment	count_of_appointments
3652	E54000034	2021-08	Attended	GP	Face-to-Face	1 Day	6553
3653	E54000034	2021-08	Attended	GP	Face-to-Face	15 to 21 Days	2390
3654	E54000034	2021-08	Attended	GP	Face-to-Face	2 to 7 Days	10547
3655	E54000034	2021-08	Attended	GP	Face-to-Face	22 to 28 Days	937
3656	E54000034	2021-08	Attended	GP	Face-to-Face	8 to 14 Days	4961
...	...	...	...	...	...	...	...
596816	E54000050	2022-06	Unknown	Unknown	Unknown	2 to 7 Days	21
596817	E54000050	2022-06	Unknown	Unknown	Unknown	22 to 28 Days	8
596818	E54000050	2022-06	Unknown	Unknown	Unknown	8 to 14 Days	28
596819	E54000050	2022-06	Unknown	Unknown	Unknown	More than 28 Days	17
596820	E54000050	2022-06	Unknown	Unknown	Unknown	Same Day	10

223418 rows × 7 columns

6.b.

In [714]:
```
# Create an aggregated data set to review the different features.
ar_agg = ar_2.groupby(['appointment_month', 'hcp_type', 'appointment_status',
                       'appointment_mode', 'time_between_book_and_appointment']).sum().reset_index()

# View the DataFrame.
ar_agg
```

Out[714]:

	appointment_month	hcp_type	appointment_status	appointment_mode	time_between_book_and_appointment	count_of_appointments
0	2021-08	GP	Attended	Face-to-Face	1 Day	507835
1	2021-08	GP	Attended	Face-to-Face	15 to 21 Days	194726
2	2021-08	GP	Attended	Face-to-Face	2 to 7 Days	959486
3	2021-08	GP	Attended	Face-to-Face	22 to 28 Days	102111
4	2021-08	GP	Attended	Face-to-Face	8 to 14 Days	398772
...	...	...	...	...	...	...
3749	2022-06	Unknown	Unknown	Unknown	8 to 14 Days	5494
3750	2022-06	Unknown	Unknown	Unknown	More than 28 Days	5115
3751	2022-06	Unknown	Unknown	Unknown	Same Day	1914
3752	2022-06	Unknown	Unknown	Unknown	Unknown / Data Quality	53
3753	2022-06	Unknown	Unknown	Video/Online	2 to 7 Days	1

3754 rows × 6 columns

6.c.

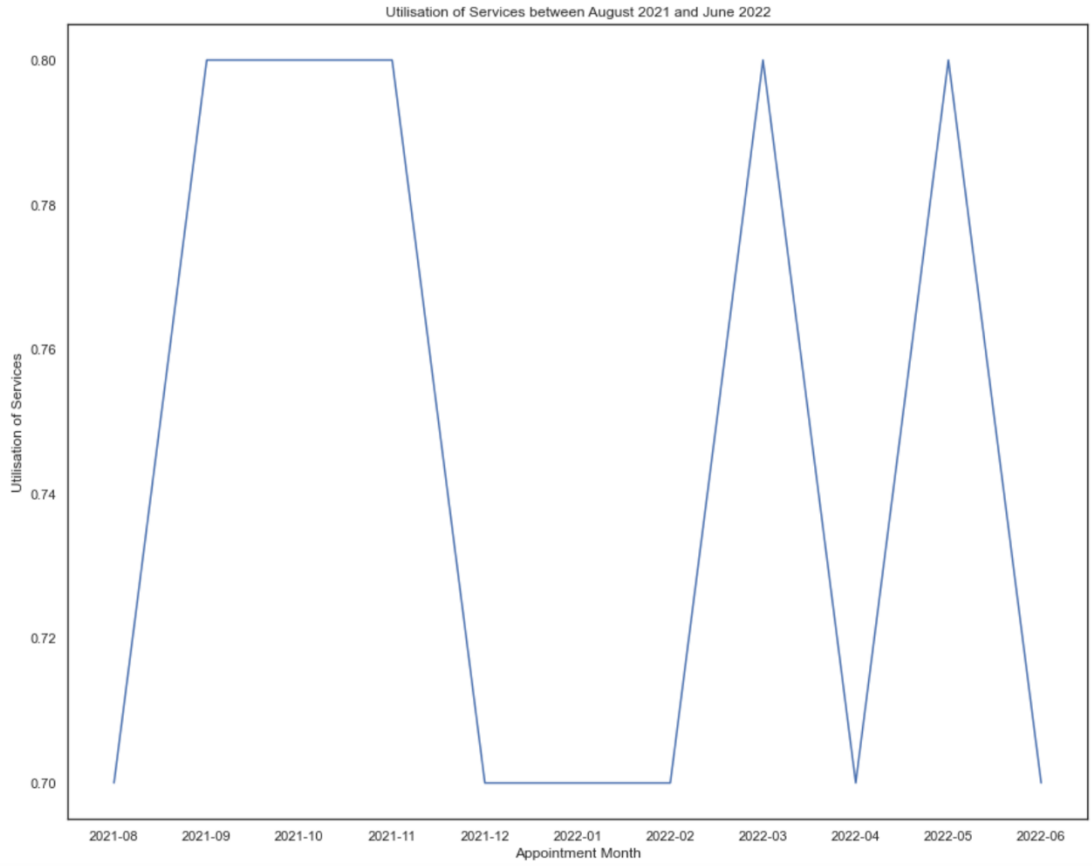
In [803]:
```
# Determine the total number of appointments per month.
ar_df = ar_agg.groupby(['appointment_month']).sum(
    ['count_of_appointments']).reset_index()
ar_df
# Add a new column to indicate the average utilisation of services.
# Monthly aggregate / 30 to get to a daily value.
ar_df['avg_daily_app'] = (ar_df['count_of_appointments']/30).round(1)
ar_df['utilisation'] = (ar_df['avg_daily_app']/1200000).round(1)
# View the DataFrame

ar_df['utilisation'] = (ar_df['avg_daily_app']/1200000).round(1)
ar_df = ar_df[['appointment_month', 'count_of_appointments',
               'avg_daily_app', 'utilisation']]
ar_df
```

Out[803]:

	appointment_month	count_of_appointments	avg_daily_app	utilisation
0	2021-08	23852171	795072.4	0.7
1	2021-09	28522501	950750.0	0.8
2	2021-10	30303834	1010127.8	0.8
3	2021-11	30405070	1013502.3	0.8
4	2021-12	25140776	838025.9	0.7
5	2022-01	25635474	854515.8	0.7
6	2022-02	25355260	845175.3	0.7
7	2022-03	29595038	986501.3	0.8
8	2022-04	23913060	797102.0	0.7
9	2022-05	27495508	916516.9	0.8
10	2022-06	25828078	860935.9	0.7

6.d.

In [673]:
```
# Plot monthly capacity utilisation.
# Create a lineplot.
utilisation_ax = sns.lineplot(
    data=ar_df, x='appointment_month', y='utilisation', ci=None)
# Customise and tidy the plot
utilisation_ax.set_xlabel('Appointment Month')
utilisation_ax.set_ylabel('Utilisation of Services')
utilisation_ax.set_title(
    "Utilisation of Services between August 2021 and June 2022")
```



7.a.

In [806]:
```
# Create a new to isolate the DNA appointment status
dna = ar_agg[ar_agg['appointment_status']=='DNA']

# View the DataFrame
dna.head()
```

Out[806]:

	appointment_month	hcp_type	appointment_status	appointment_mode	time_between_book_and_appointment	count_of_appointments
40	2021-08	GP	DNA	Face-to-Face	1 Day	16314
41	2021-08	GP	DNA	Face-to-Face	15 to 21 Days	12960
42	2021-08	GP	DNA	Face-to-Face	2 to 7 Days	43754
43	2021-08	GP	DNA	Face-to-Face	22 to 28 Days	7457
44	2021-08	GP	DNA	Face-to-Face	8 to 14 Days	24346

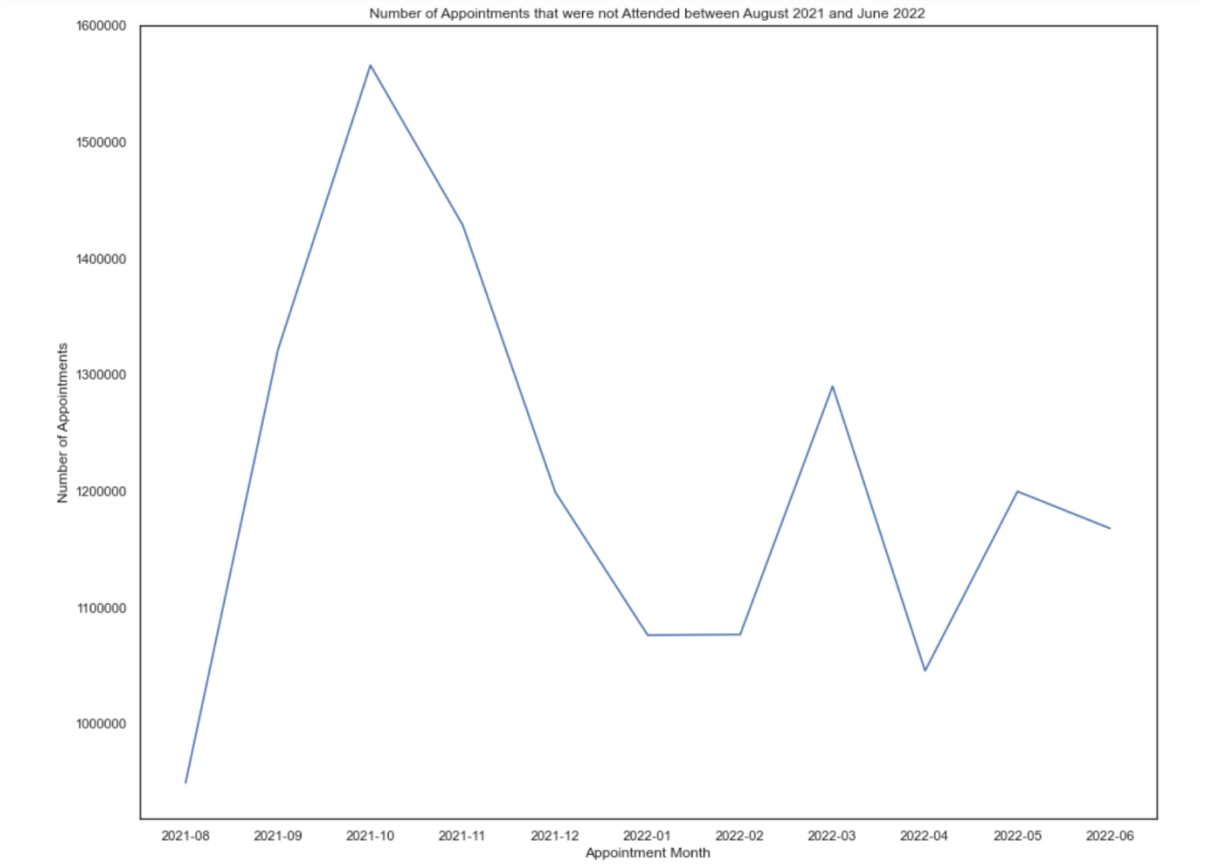
In [946]:
```
# Create a DataFrame to group the values for DNA
dna_group = dna.groupby(['appointment_month']).sum().reset_index()
dna_group
```

Out[946]:

	appointment_month	count_of_appointments
0	2021-08	949137
1	2021-09	1321348
2	2021-10	1565624
3	2021-11	1428087
4	2021-12	1198866
5	2022-01	1076013
6	2022-02	1076658
7	2022-03	1289888
8	2022-04	1045455
9	2022-05	1199518
10	2022-06	1167790

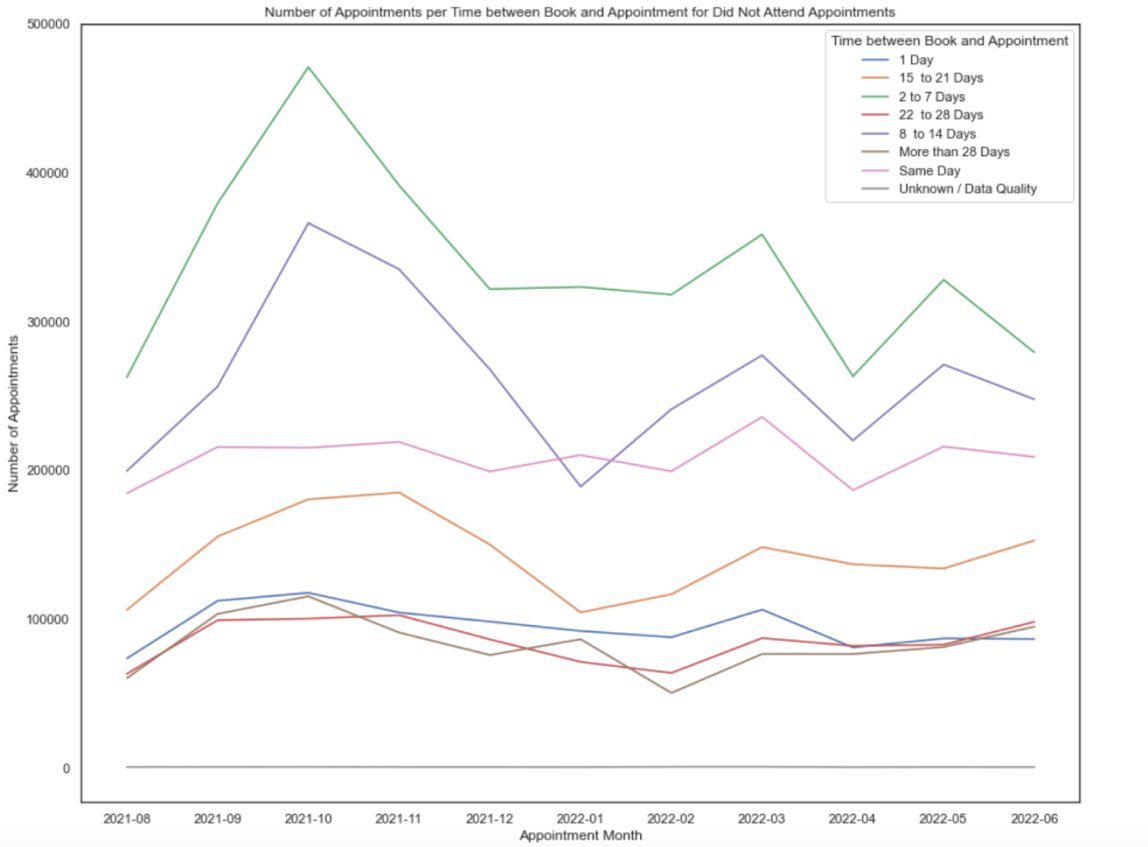
7.b.

In [950]:
```
# Create a line plot to show number of DNA appointments.
dna_ax = sns.lineplot(x='appointment_month', y='count_of_appointments', data=dna_group, ci=None)

dna_ax.set_xlabel('Appointment Month')
dna_ax.set_ylabel('Number of Appointments')
dna_ax.set_title(
    "Number of Appointments that were not Attended between August 2021 and June 2022")

dna_ax.set_yticks([1000000, 1100000, 1200000, 1300000, 1400000, 1500000, 1600000])
dna_ax.set_yticklabels(['1000000', '1100000', '1200000', '1300000', '1400000', '1500000', '1600000'])
```



7.c.

In [990]:
```
dna_time_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',\
             hue='time_between_book_and_appointment', data=dna_time, ci=None)

# Customise and tidy the graph
dna_time_ax.legend(title='Time between Book and Appointment', loc='upper right')
dna_time_ax.set_xlabel('Appointment Month')
dna_time_ax.set_ylabel('Number of Appointments')
dna_time_ax.set_title(
    "Number of Appointments per Time between Book and Appointment for Did Not Attend Appointments")
dna_time_ax.set_yticks([0, 100000, 200000, 300000, 400000, 500000])
dna_time_ax.set_yticks([50000, 150000, 250000, 350000,
                        450000], minor=True)
dna_time_ax.set_yticklabels(['0', '100000', '200000', '300000',
                             '400000', '500000'])
```



7.d.

In [808]:
```
# Create a new subset on the 8 to 14 Days time between book and appointment
eight_fourteen = ar_agg[ar_agg['time_between_book_and_appointment']
                        == '8  to 14 Days']

# View the output
eight_fourteen.head()
```

Out[808]:

	appointment_month	hcp_type	appointment_status	appointment_mode	time_between_book_and_appointment	count_of_appointments
4	2021-08	GP	Attended	Face-to-Face	8 to 14 Days	398772
12	2021-08	GP	Attended	Home Visit	8 to 14 Days	1060
20	2021-08	GP	Attended	Telephone	8 to 14 Days	488939
28	2021-08	GP	Attended	Unknown	8 to 14 Days	7736
36	2021-08	GP	Attended	Video/Online	8 to 14 Days	3561

In [761]:
```
# From the new output - groupby hcp_type for 8 to 14 Days time between book and appointment
type_8_14 = eight_fourteen.groupby(['appointment_month', 'hcp_type']).sum().reset_index()

# View the output
type_8_14.head()
```

Out[761]:

	appointment_month	hcp_type	count_of_appointments
0	2021-08	GP	975628
1	2021-08	Other Practice staff	1867275
2	2021-08	Unknown	46247
3	2021-09	GP	1253666
4	2021-09	Other Practice staff	2044882

7.e.

In [810]:
```
# Create a new subset on the 2 to 7 Days time between book and appointment
two_seven = ar_agg[ar_agg['time_between_book_and_appointment']
                   == '2 to 7 Days']

# View the output
two_seven
```

Out[810]:

	appointment_month	hcp_type	appointment_status	appointment_mode	time_between_book_and_appointment	count_of_appointments
2	2021-08	GP	Attended	Face-to-Face	2 to 7 Days	959486
10	2021-08	GP	Attended	Home Visit	2 to 7 Days	7012
18	2021-08	GP	Attended	Telephone	2 to 7 Days	975156
26	2021-08	GP	Attended	Unknown	2 to 7 Days	20961
34	2021-08	GP	Attended	Video/Online	2 to 7 Days	13169
...	...	...	...	...	...	...
3723	2022-06	Unknown	Unknown	Face-to-Face	2 to 7 Days	3146
3731	2022-06	Unknown	Unknown	Home Visit	2 to 7 Days	984
3739	2022-06	Unknown	Unknown	Telephone	2 to 7 Days	1000
3747	2022-06	Unknown	Unknown	Unknown	2 to 7 Days	4929
3753	2022-06	Unknown	Unknown	Video/Online	2 to 7 Days	1

484 rows × 6 columns

In [811]:
```
# From the new output - groupby hcp_type for 2 to 7 Days time between book and appointment
type_2_7 = two_seven.groupby(['appointment_month', 'hcp_type']).sum().reset_index()

# View the output
type_2_7.head()
```

Out[811]:

	appointment_month	hcp_type	count_of_appointments
0	2021-08	GP	2107045
1	2021-08	Other Practice staff	2709478
2	2021-08	Unknown	108437
3	2021-09	GP	2549886
4	2021-09	Other Practice staff	3276358

7.f.

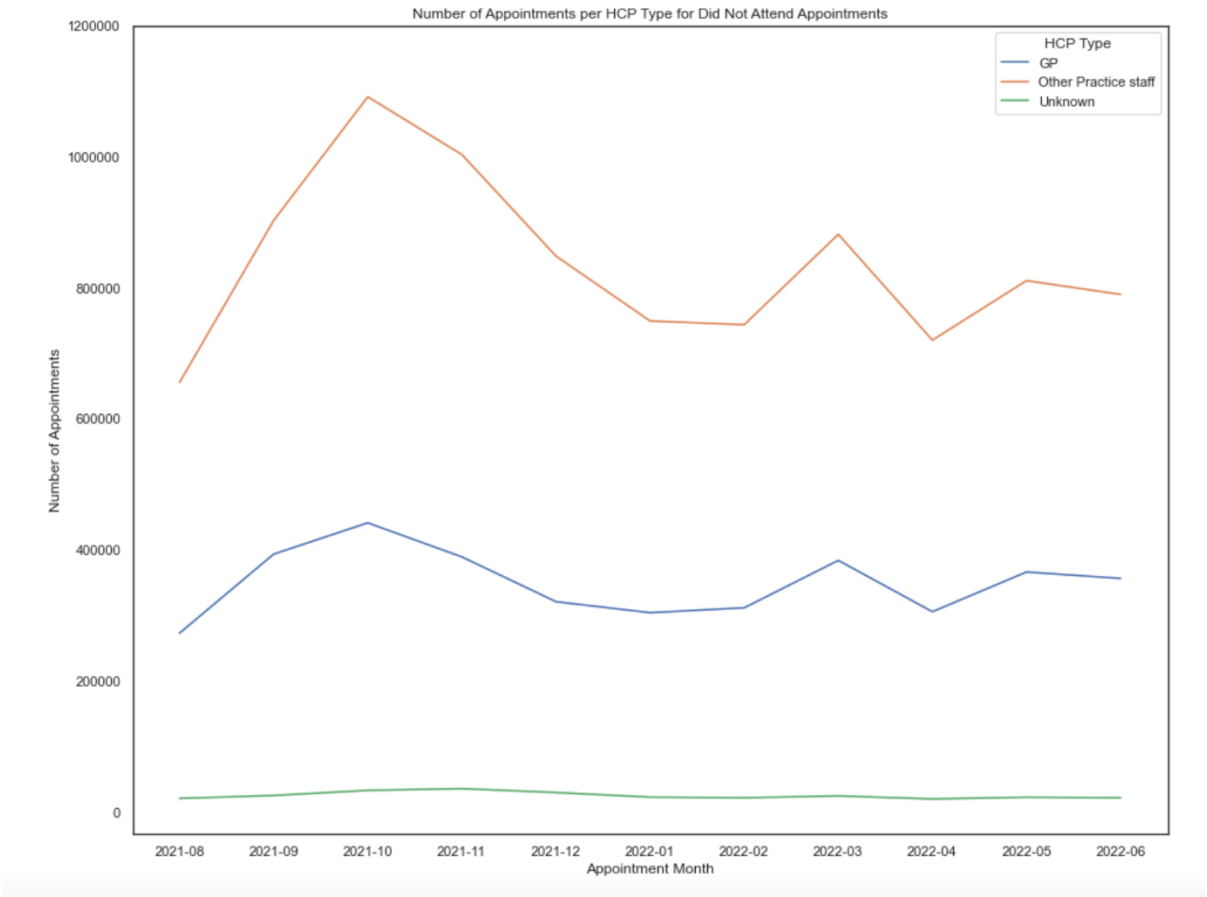
In [812]:
```
# Plot a lineplot to show the relationship of hcp for 8 to 14 days
eight_fourteen_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',\
             hue='hcp_type', data=type_8_14, ci=None)

# Customise and tidy the graph
eight_fourteen_ax.legend(title='Appointment Mode', bbox_to_anchor=(1, 1))
eight_fourteen_ax.set_xlabel('Appointment Month')
eight_fourteen_ax.set_ylabel('Number of Appointments')
eight_fourteen_ax.set_title(
    "Number of Appointments per HCP Type for 8 to 14 Days Time Between Book and Appointment")
eight_fourteen_ax.set_yticks([0, 500000, 1000000, 1500000, 2000000, 2500000, 3000000])
eight_fourteen_ax.set_yticks([250000, 750000, 1250000, 1750000,
                      2250000, 2750000], minor=True)
eight_fourteen_ax.set_yticklabels(['0', '500000', '1000000', '1500000', '2000000', '2500000',
                          '3000000'])
```

7.g.

In [813]:
```
# Plot a lineplot to show the relationship of hcp for 2 to 7 days
two_seven_ax = sns.lineplot(x='appointment_month', y='count_of_appointments',\
            hue='hcp_type', data=type_2_7, ci=None)


# Customise and tidy the graph
two_seven_ax.legend(title='Appointment Mode', bbox_to_anchor=(1, 1))
two_seven_ax.set_xlabel('Appointment Month')
two_seven_ax.set_ylabel('Number of Appointments')
two_seven_ax.set_title(
    "Number of Appointments per HCP Type for 8 to 14 Days Time Between Book and Appointment")
two_seven_ax.set_yticks([0, 500000, 1000000, 1500000, 2000000, 2500000, 3000000, 3500000, 4000000])
two_seven_ax.set_yticks([250000, 750000, 1250000, 1750000,
                         2250000, 2750000, 3250000, 3750000], minor=True)
two_seven_ax.set_yticklabels(['0', '500000', '1000000', '1500000', '2000000', '2500000',
                              '3000000', '3500000', '4000000'])
```

7.h.

In [792]:
```
dna_type_ax = sns.lineplot(
    x='appointment_month', y='count_of_appointments', hue='hcp_type', data=dna_type, ci=None)

# Customise and tidy the graph
dna_type_ax.legend(title='HCP Type', loc='upper right')
dna_type_ax.set_xlabel('Appointment Month')
dna_type_ax.set_ylabel('Number of Appointments')
dna_type_ax.set_title(
    "Number of Appointments per HCP Type for Did Not Attend Appointments")
dna_type_ax.set_yticks([0, 200000, 400000, 600000, 800000, 1000000, 1200000])
dna_type_ax.set_yticks([100000, 300000, 500000, 700000,
                        900000, 1100000], minor=True)
dna_type_ax.set_yticklabels(['0', '200000', '400000', '600000',
                             '800000', '1000000', '1200000'])
```


Number of Appointments per HCP Type for Did Not Attend Appointments
HCP Type
GP
Other Practice staff
Unknown
1200000
1000000
800000
600000
400000
200000
0
Number of Appointments
2021-08
2021-09
2021-10
2021-11
2021-12
2022-01
2022-02
2022-03
2022-04
2022-05
2022-06
Appointment Month

7.i.

In [983]:
```
# Create a new DataFrame to isolate Other Practice Staff HCP Type
ops_ar = ar_agg[ar_agg['hcp_type'] == 'Other Practice staff']
# Filter new DataFrame futher to group appointments for Other Practice Type
ops_ar_group = ops_ar.groupby(['appointment_month', 'hcp_type']).sum().reset_index()
ops_ar_group
```

Out[983]:

	appointment_month	hcp_type	count_of_appointments
0	2021-08	Other Practice staff	10797821
1	2021-09	Other Practice staff	13126731
2	2021-10	Other Practice staff	14942504
3	2021-11	Other Practice staff	14432800
4	2021-12	Other Practice staff	11614470
5	2022-01	Other Practice staff	11705142
6	2022-02	Other Practice staff	11688107
7	2022-03	Other Practice staff	13528583
8	2022-04	Other Practice staff	11217738
9	2022-05	Other Practice staff	12823200
10	2022-06	Other Practice staff	12105677

In [985]:
```
# Create a new data frame to focus on Other Practice Staff DNA appointments
ops_dna = dna_type[dna_type['hcp_type'] == 'Other Practice staff']
# Rename column for future merge
ops_dna.rename(columns={'count_of_appointments':'count_dna_appointments'}, inplace=True)
# Merge two df
ops_dna = pd.merge(ops_dna, ops_ar_group[['appointment_month','count_of_appointments']]
                   ,on='appointment_month', how='left')
# Calculate the percentage appointments that are not attended
ops_dna['percentage_dna'] = ((ops_dna['count_dna_appointments']/ops_dna['count_of_appointments'])*100).round(2)
ops_dna
```

Out[985]:

	appointment_month	hcp_type	appointment_status	count_dna_appointments	count_of_appointments	percentage_dna
0	2021-08	Other Practice staff	DNA	655689	10797821	6.07
1	2021-09	Other Practice staff	DNA	902812	13126731	6.88
2	2021-10	Other Practice staff	DNA	1091576	14942504	7.31
3	2021-11	Other Practice staff	DNA	1003396	14432800	6.95
4	2021-12	Other Practice staff	DNA	848447	11614470	7.31
5	2022-01	Other Practice staff	DNA	749418	11705142	6.40
6	2022-02	Other Practice staff	DNA	743623	11688107	6.36
7	2022-03	Other Practice staff	DNA	881640	13528583	6.52
8	2022-04	Other Practice staff	DNA	720150	11217738	6.42
9	2022-05	Other Practice staff	DNA	810955	12823200	6.32
10	2022-06	Other Practice staff	DNA	789911	12105677	6.53