NHS England

TECHNICAL REPORT

Using Python with NHS Data

Anya Chan

Contents

Overview

This report examines how digital transformation, driven by the COVID-19 pandemic, impacted the healthcare sector, particularly NHS GP services in England. The pandemic caused 205,540 deaths and 24,315,983 infections in the UK (United Kingdom COVID - Coronavirus Statistics, n.d.), accelerating the adoption of digital solutions.

The focus is on NHS GP services from January 2020 to June 2022, aiming to improve their efficiency moving forward. The main questions explored are:

1. Has there been adequate staff and capacity in the networks?
2. What was the actual utilisation of resources?

Analytical approach

The analysis began by importing and exploring the data to identify key questions that could be answered using Python. Key libraries like Pandas, NumPy, Matplotlib, and Seaborn were imported into Jupyter Notebook. An Excel file containing NHS ICB codes and regions was created and merged with other relevant data, such as NHS England Names and Codes for 2025.

New data frames—location_ar, location_nc, and location_ad—were created. During the analysis, it was found that the gp-reg-pat-prac-map dataset had three missing icb_ons_code values, which were due to inactive entries from practices in the South East region.

```
# Create a dataframe for the sub_icb_location_code with Region Name for appointment duration

location_ad = pd.merge ( ad, location, how ='left', on = 'icb_ons_code')
location_ad
```

```
# Create a dataframe for the sub_icb_location_code with Region Name for national categories
location = pd.read_csv('location.csv')
location.head()

location_nc = pd.merge ( nc, location, how ='left', on = 'icb_ons_code')
location_nc
```

appointment_date	icb_ons_code	sub_icb_location_name	service_setting	context_type	national_category	count_of_appointments	appointment_month	REGION_NAME
2021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	Primary Care Network	Care Related Encounter	Patient contact during Care Home Round	3	2021-08	North East and Yorkshire
		NHS North East and		Care Related				North East and

21,604 duplicate records were found in the appointments regional data frame, but they were kept since they mostly consisted of repeated dates. Removing them could have affected the results, especially since the data covers a long period.

```
# Determine the maximum and minimum dates in the ad DataFrame.
# Use appropriate docstrings.
ad['appointment_date'].agg(['min', 'max'])
```

```
min    01-Apr-22
max    31-May-22
Name: appointment_date, dtype: object
```

```
# Determine the maximum and minimum dates in the ad DataFrame.
# Use appropriate docstrings.
ar['appointment_month'].agg(['min', 'max'])
```

```
min    2020-01
max    2022-06
Name: appointment_month, dtype: object
```

```
# Determine the minimum and maximum dates in the nc DataFrame.
# Use appropriate docstrings.
nc['appointment_date'].agg(['min', 'max'])
```

```
min   2021-08-01
max   2022-06-30
Name: appointment_date, dtype: datetime64[ns]
```

Date formats in the nc, ar, and ad datasets were standardized for consistency.

```
# Convert to datetime obviously if not already in correct format
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'])
ar['year'] = ar['appointment_month'].dt.year
ar['month'] = ar['appointment_month'].dt.month_name()
# View the output
print(ar['appointment_month'])
```

```
# Change the date format of ar['appointment_date'].
ar['appointment_month'] = ar['appointment_month'].dt.strftime("%m/%d/%Y")
# View the DateFrame.
ar['appointment_month'].head(5)
```

```
# Change the date format of nc['appointment_date'].
nc['appointment_date'] = pd.to_datetime(nc['appointment_date'])
nc['appointment_date'] = nc['appointment_date'].dt.strftime("%m/%d/%Y")
# View the DateFrame.
nc['appointment_date'].head(5)
```

The nc data frame was used for daily trend analysis, while the ar data frame focused on monthly trends. Regional differences were examined by grouping data based on various factors to count the total number of appointments in each context. To measure resource usage, the number of appointments was compared to the target of 1.2 million daily appointments from the Week 6 activity.

Finally, Twitter data was analyzed to identify the most discussed topics, focusing on those mentioned more than 20 times. Various questions were explored to understand the factors affecting appointment attendance and trends in GP service usage.

	#text	count_of_hashtag_text	group
0	#healthcare	716	Health
1	#health	80	Health
2	#medicine	41	Health
3	#ai	40	Tech
4	#job	38	Job
5	#medical	35	Health
6	#strategy	30	Strategy
7	#pharmaceutical	28	Health
8	#pharma	25	Health
9	#marketing	25	Job
10	#digitalhealth	25	Tech
11	#biotech	24	Health
12	#medtwitter	24	Health
13	#competitiveintelligence	24	Job
14	#meded	23	Health

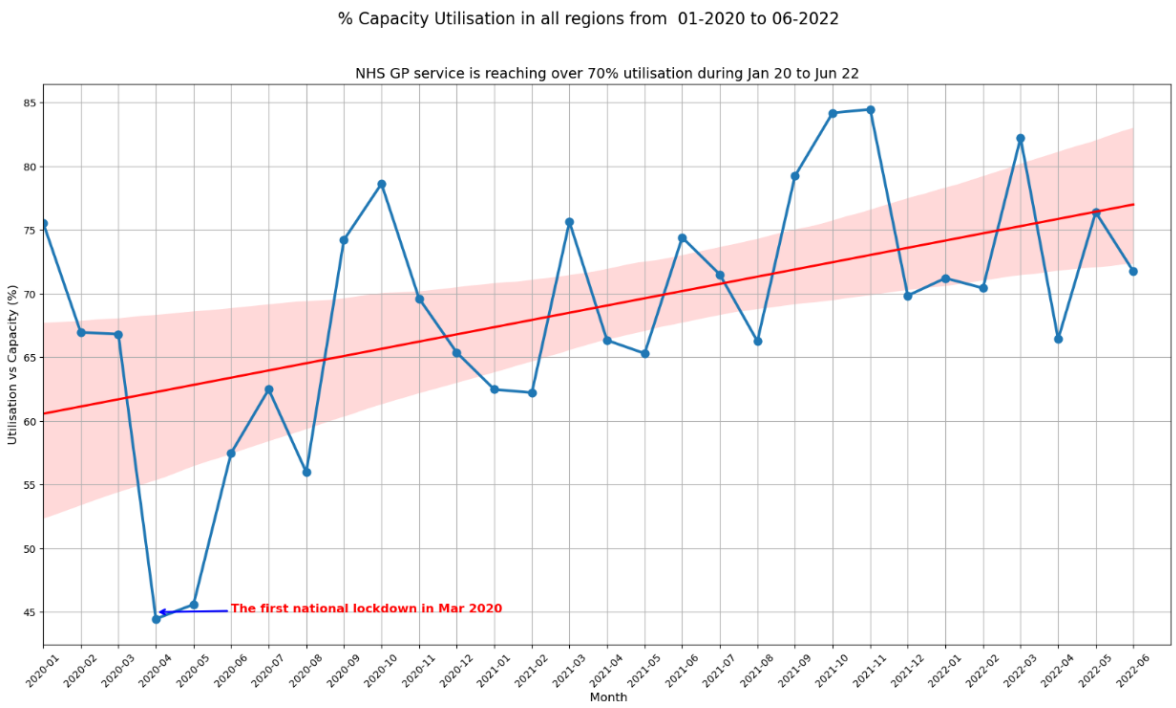
```
group_mapping = {
    '#health': 'Health',
    '#healthcare': 'Health',
    '#healthcare': 'Health',
    '#medicalcare': 'Health',
    '#covid': 'Covid',
    '#covid19': 'Covid',
    '#mental health': 'Health',
    '#medicine' : 'Health',
    '#medicine': 'Health',
    '#ai' : 'Tech',
    '#job' : 'Job',
    '#medical' : 'Health',
    '#strategy' : 'Strategy',
    '#pharmaceutical' : 'Health',
    '#pharma': 'Health',
    '#digitalhealth' : 'Tech',
    '#marketing' :'Job',
    '#biotech': 'Health',
    '#medtwitter': 'Health',
    '#competitiveintelligence' :'Job',
    '#meded' :'Health'}
filtered_tags_df['group'] = filtered_tags_df['#text'].map(group_mapping)
filtered_tags_df
```

Different questions were posed to analyse the various factors influencing appointment attendance and the trends in patients utilizing GP services.

Data Visualisation:

Should NHS hire more staff? The capacity utilisation:

A point plot chart was created using the ar dataset to visualize NHS utilisation versus capacity. This helped assess whether NHS should consider increasing staff levels. The maximum capacity of 1.2 million appointments per day was used, and the 'utilisation vs capacity%' was calculated using the following code:



(A trend line was added to indicate a continued increase, and the annotate function highlighted the sharp drop in March 2020 due to the first national lockdown in the UK.)The maximum capacity of 1.2 million appointments per day was used, and the 'utilisation vs capacity%' was calculated using the following code in appendixes to created ar_df to plot the line chart.

ar_df

appointment_month	count_of_appointments	utilisation	utilisation_vs_capacity%
2020-01	27199296	9.066432e+05	75.553600
2020-02	24104621	8.034874e+05	66.957281
2020-03	24053468	8.017823e+05	66.815189
2020-04	16007881	5.335960e+05	44.466336
2020-05	16417212	5.472404e+05	45.603367
2020-06	20690805	6.896935e+05	57.474458
2020-07	22491437	7.497146e+05	62.476214
2020-08	20150520	6.716840e+05	55.973667
2020-09	26714255	8.904752e+05	74.206264
2020-10	28301932	9.433977e+05	78.616478
2020-11	25061602	8.353867e+05	69.615561
2020-12	23535936	7.845312e+05	65.377600
2021-01	22492069	7.497356e+05	62.477969
2021-02	22399569	7.466523e+05	62.221025

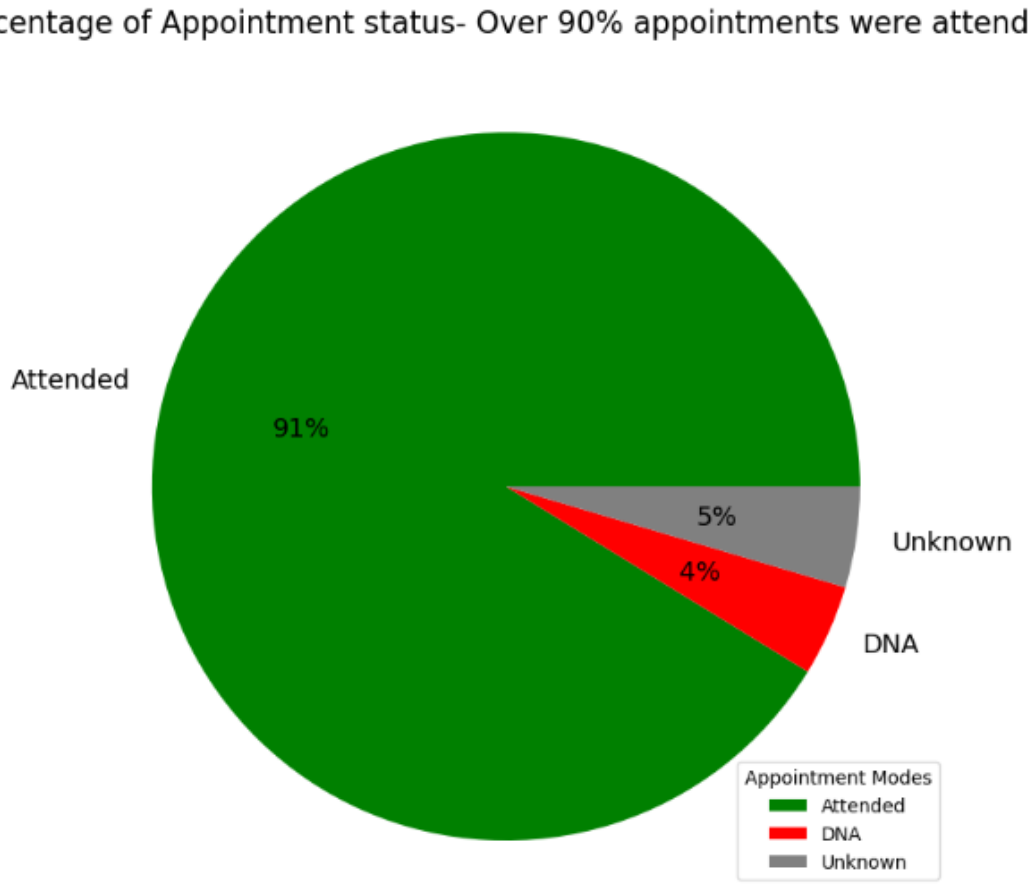
What percentage of appointments have been attended?

To determine the percentage of attended appointments, I created a new data frame called appointment_mode_counts by grouping the data by appointment_status.

```
: appointment_mode_counts
```

```
: appointment_status
  Attended    677755876
  DNA          30911233
  Name: count_of_appointments, dtype: int64
```

Then used a pie chart to visualize the results. It shows that 91% of appointments were attended, 4% were missed (DNA), and 5% had an unknown status.



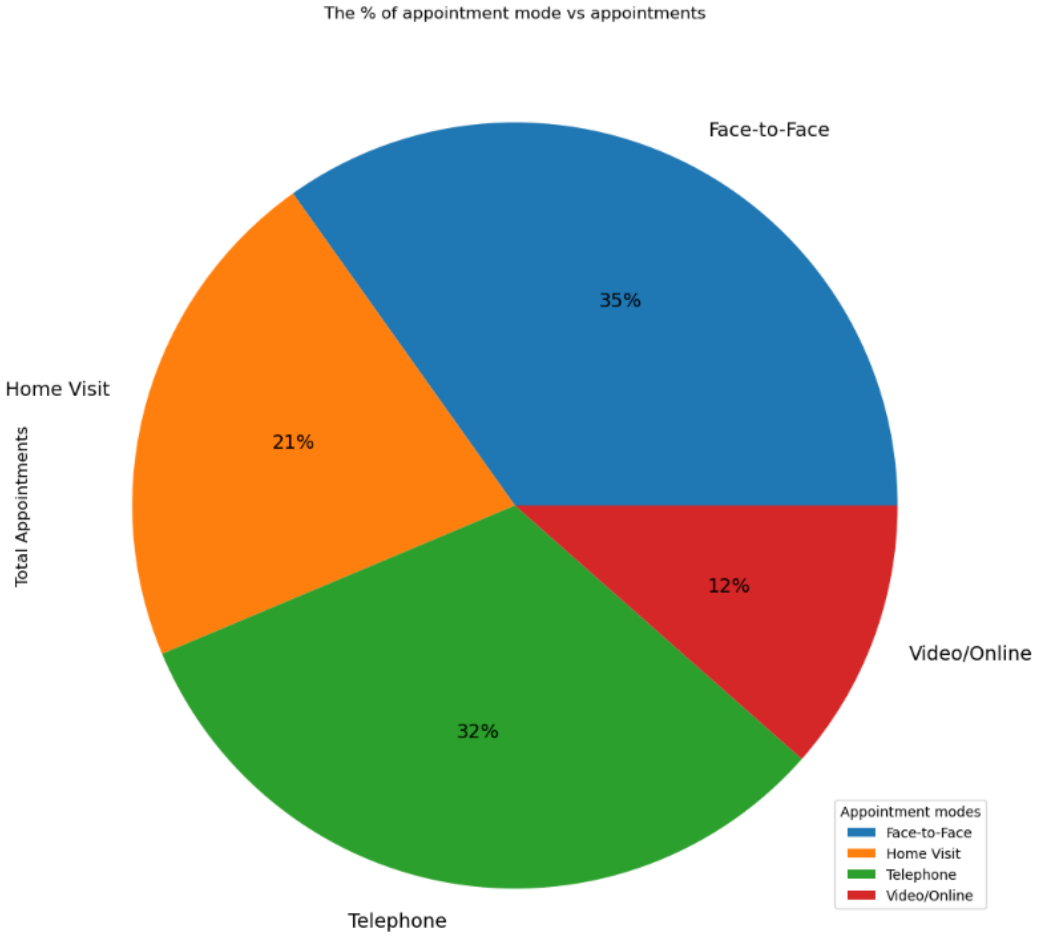
Which appointment mode is the most popular?

To visualize the most used appointment modes, created ar_mode_clean to exclude "Unknown" data.

```
# Create a ar dataframe to exclude "Unknown" in appointment mode
ar_mode_clean =  ar[ar['appointment_mode'] != "Unknown"]
ar_mode_clean
```

	icb_ons_code	appointment_month	appointment_status	hcp_type	appointment_mode	time_between_book_and_appointment	count_of_appointments
0	E54000034	2020-01	Attended	GP	Face-to-Face	1 Day	8107
1	E54000034	2020-01	Attended	GP	Face-to-Face	15 to 21 Days	6791
2	E54000034	2020-01	Attended	GP	Face-to-Face	2 to 7 Days	20686
3	E54000034	2020-01	Attended	GP	Face-to-Face	22 to 28 Days	4268
4	E54000034	2020-01	Attended	GP	Face-to-Face	8 to 14 Days	11971
...	...	...	...	...	...	...	...
596809	E54000050	2022-06	Unknown	Unknown	Telephone	15 to 21 Days	1
596810	E54000050	2022-06	Unknown	Unknown	Telephone	2 to 7 Days	16

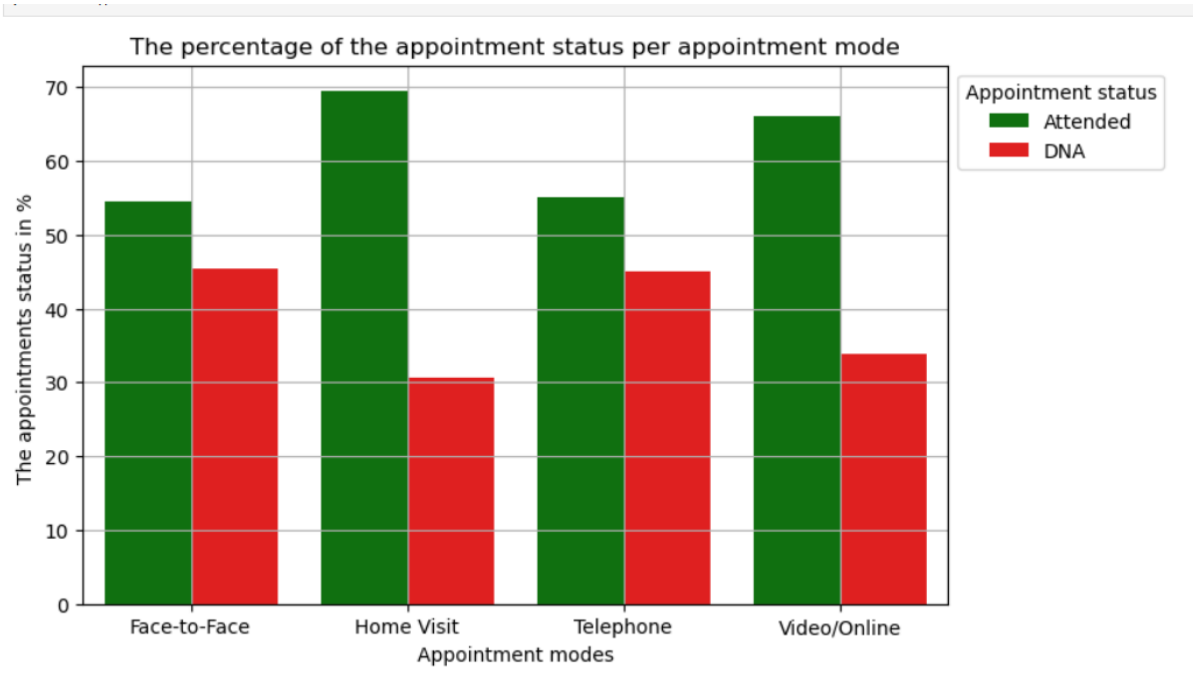
A pie chart was made to show the percentage of each appointment mode. The chart revealed that most appointments are Face to Face (35%), followed by Telephone (32%).



To analyse this further, the attendance rate for each appointment mode was examined using the ar dataset. A table was created to compare appointment status across different modes.

	appointment_mode	appointment_status	count	total_count	percentage
0	Face-to-Face	Attended	64478	118248	54.53
1	Face-to-Face	DNA	53770	118248	45.47
2	Home Visit	Attended	48608	69994	69.45
3	Home Visit	DNA	21386	69994	30.55
4	Telephone	Attended	60796	110492	55.02
5	Telephone	DNA	49696	110492	44.98
6	Video/Online	Attended	28729	43453	66.12
7	Video/Online	DNA	14724	43453	33.88

Face-to-Face was the most common appointment mode, but only 54.53% of patients attended. In contrast, Video/Online appointments had a higher attendance rate at 66.12%, while Home Visits had the highest at 69.45%.



How did Covid-19 change the preferred appointment mode?

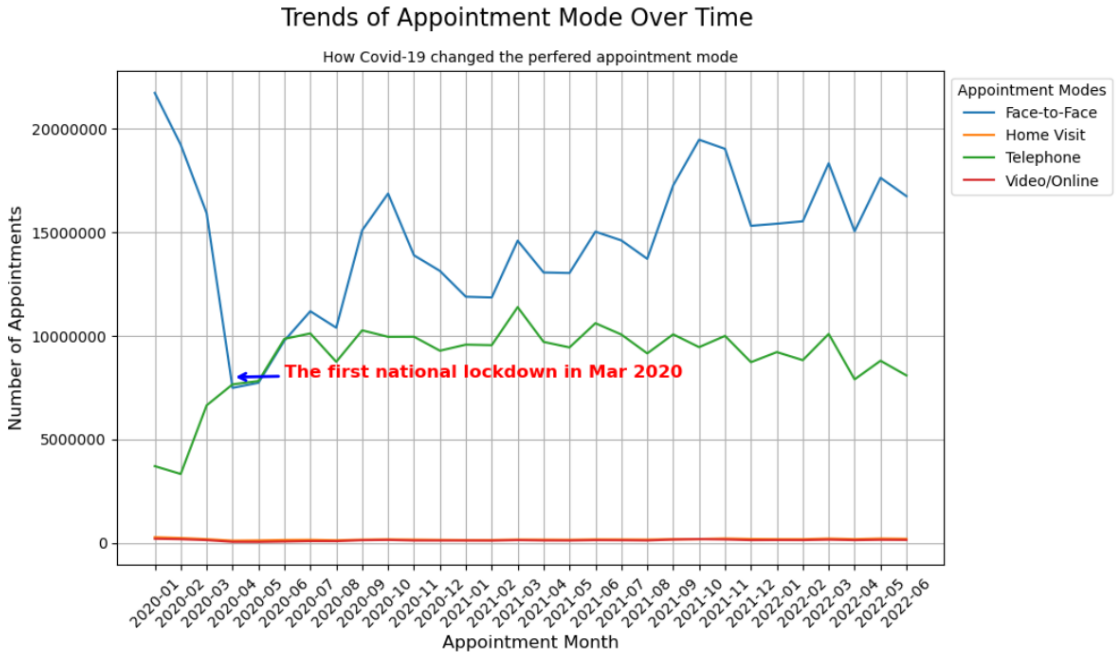
To analyse this, the ar Data Frame was used to examine appointment modes from January 2020 to June 2022.By reviewing appointment trends over this period, the data highlights shift in patient preferences due to the pandemic.

ar_df_mode

	appointment_month	appointment_mode	count_of_appointments
0	2020-01	Face-to-Face	21733394
1	2020-01	Home Visit	266942
2	2020-01	Telephone	3701775
3	2020-01	Video/Online	194206
4	2020-02	Face-to-Face	19230573
...	...	...	...
115	2022-05	Video/Online	144188
116	2022-06	Face-to-Face	16744191
117	2022-06	Home Visit	187640
118	2022-06	Telephone	8082270
119	2022-06	Video/Online	136117

To visualize the trend of appointment modes over time, a line chart was used to track changes.

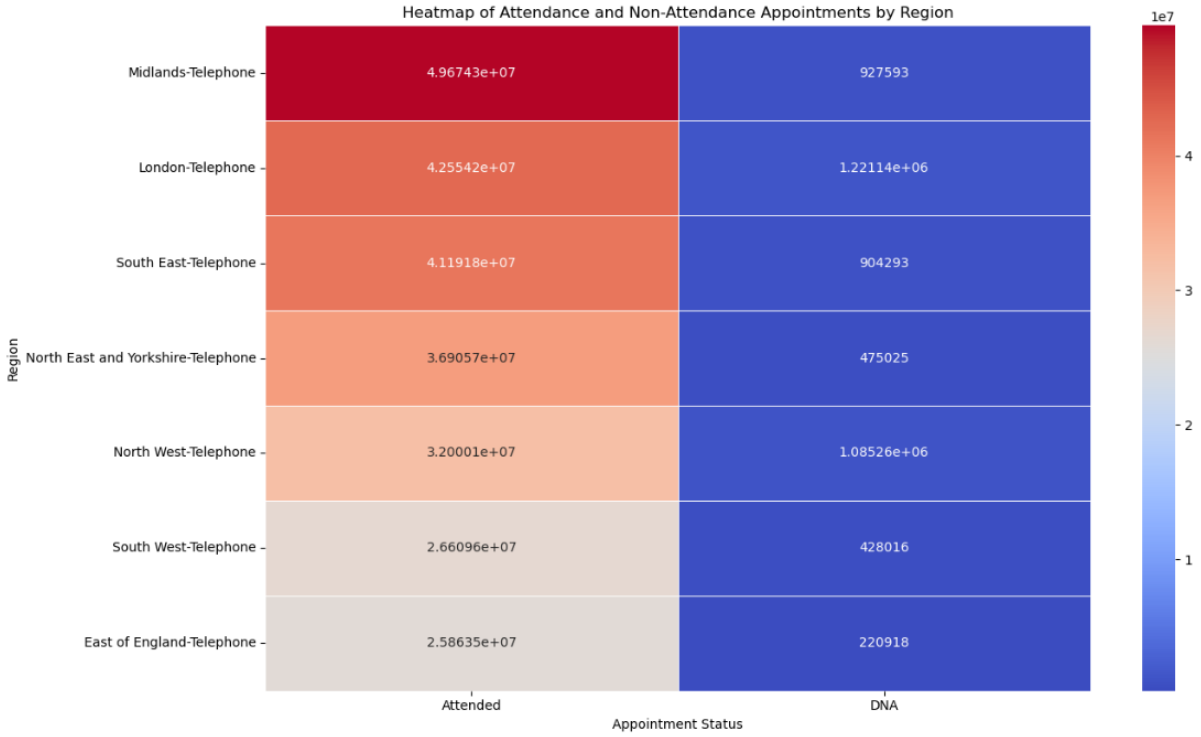
The data shows a significant shift in appointment preferences post-COVID. Telephone appointments surged from 3,701,775 in January 2020 to 6,637,656 in March 2020, coinciding with the first national lockdown. This mode remained consistently high, exceeding 7,650,000 appointments per month thereafter.



Since telephone appointments have become increasingly important post-COVID, analysing attendance rates by region offers valuable insights into how different areas have adapted to this shift.

REGION_NAME	appointment_status appointment_mode	Attended	DNA	Total Appointments
East of England	Telephone	25863545	220918	26084463
London	Telephone	42554175	1221141	43775316
Midlands	Telephone	49674266	927593	50601859
North East and Yorkshire	Telephone	36905716	475025	37380741
North West	Telephone	32000122	1085265	33085387
South East	Telephone	41191762	904293	42096055
South West	Telephone	26609640	428016	27037656

A bar chart was used to visualize the regions with the highest number of telephone appointments. The chart highlights that the Midlands has the highest number of telephone appointments (49,674,266), while the East of England has the lowest (25,863,545). Additionally, it is evident that all regions have a low number of "Did Not Attend" (DNA) appointments.



Which day of week had the most appointments?

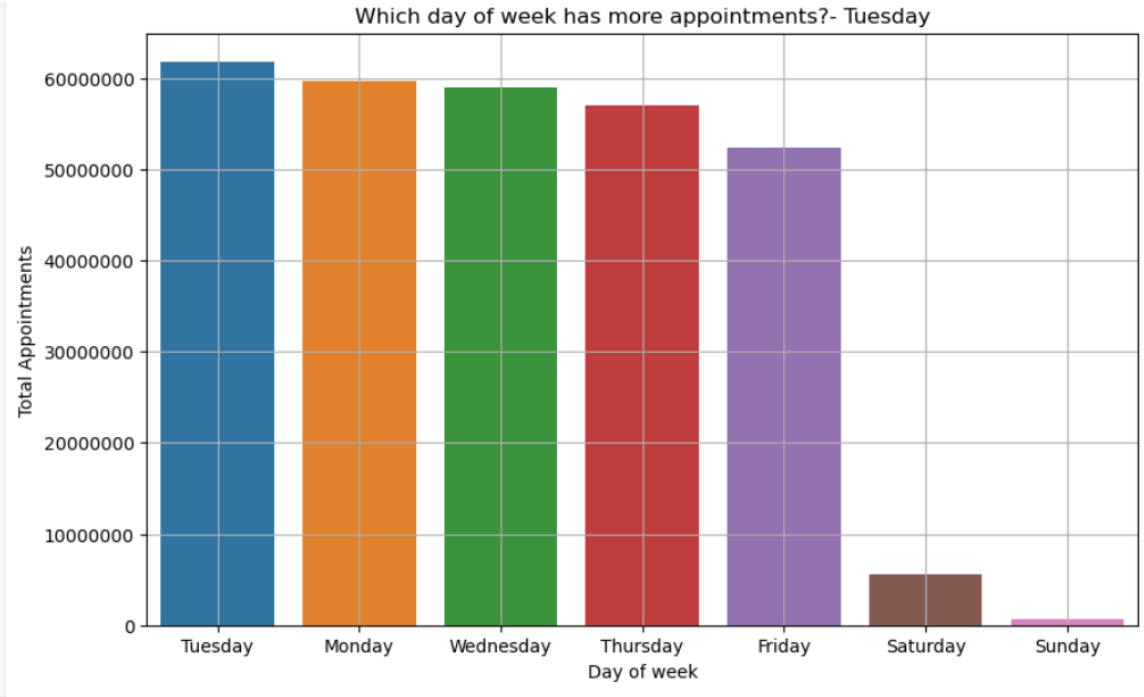
To determine which day of the week had the most appointments and help the NHS plan staffing, a "day of week" column was created using .dt.day_name().

```
]: location_nc_pivot
```

]:

	day_of_week	count_of_appointments
5	Tuesday	61806933
1	Monday	59695267
6	Wednesday	58984265
4	Thursday	56976354
0	Friday	52394868
2	Saturday	5574922
3	Sunday	614161

The analysis showed that Monday had the highest number of appointments, with most bookings happening on weekdays compared to weekends.



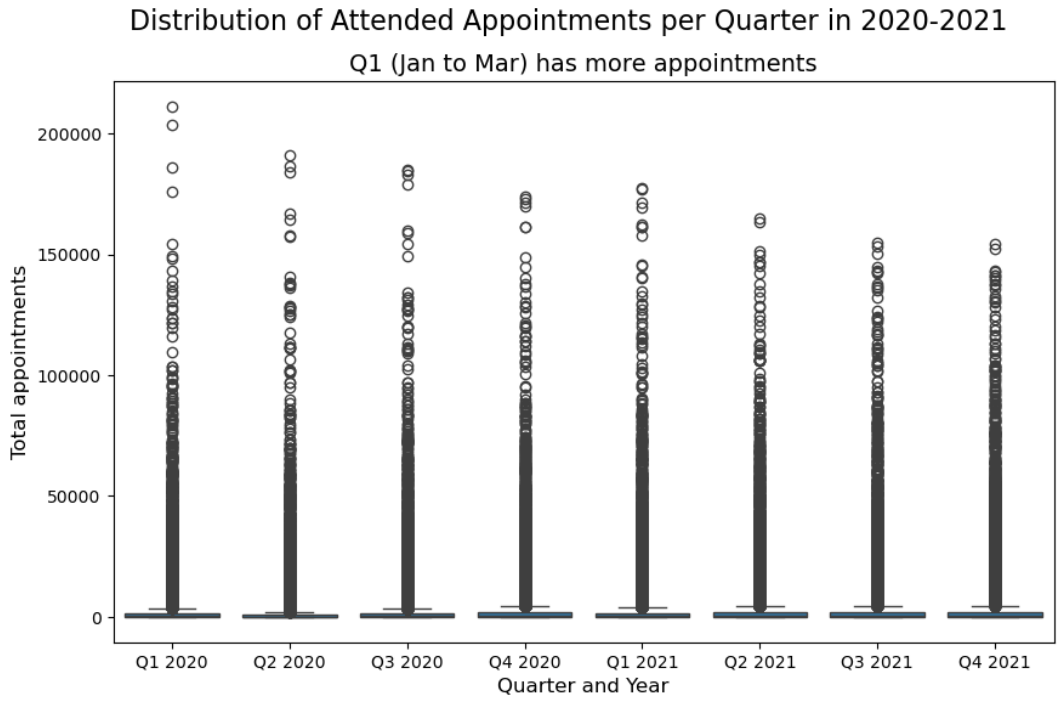
Which Seasons and Quarters Had the Most Attended Appointments?

Based on the previous analysis, it was observed that over 90% of appointments were attended. To further explore this, I created a DataFrame (combined_df) to examine the distribution of attended appointments across quarters in 2020-2021.

combined_df

ıtment_month	icb_ons_code	appointment_status	hcp_type	appointment_mode	time_between_book_and_appointment	count_of_appointments	REGION_NAME	Quarter
2020-01-01	E54000034	Attended	GP	Face-to-Face	1 Day	8107	South East	Q1 2020
2020-01-01	E54000034	Attended	GP	Face-to-Face	15 to 21 Days	6791	South East	Q1 2020
2020-01-01	E54000034	Attended	GP	Face-to-Face	2 to 7 Days	20686	South East	Q1 2020
2020-01-01	E54000034	Attended	GP	Face-to-Face	22 to 28 Days	4268	South East	Q1 2020
2020-01-01	E54000034	Attended	GP	Face-to-Face	8 to 14 Days	11971	South East	Q1 2020
...	...	...	...	...	...	...	...	...
2021-12-01	E54000050	Attended	Unknown	Unknown	2 to 7 Days	220	North East and Yorkshire	Q4 2021
2021-12-01	E54000050	Attended	Unknown	Unknown	22 to 28 Days	37	North East and Yorkshire	Q4 2021
2021-12-01	E54000050	Attended	Unknown	Unknown	8 to 14 Days	189	North East and Yorkshire	Q4 2021
2021-12-01	E54000050	Attended	Unknown	Unknown	More than 28 Days	45	North East and Yorkshire	Q4 2021
2021-12-01	E54000050	Attended	Unknown	Unknown	Same Day	1152	North East and Yorkshire	Q4 2021

From the box plot, we can see that Q1 (Jan-Mar 2020) experienced a higher volume of appointments compared to other quarters, likely due to the impact of COVID-19. However, overall, the data from 2021 shows that, once COVID-19 was under control, the highest number of appointments were still made in the **Jan-Mar** period, indicating a seasonal peak at the start of the year.

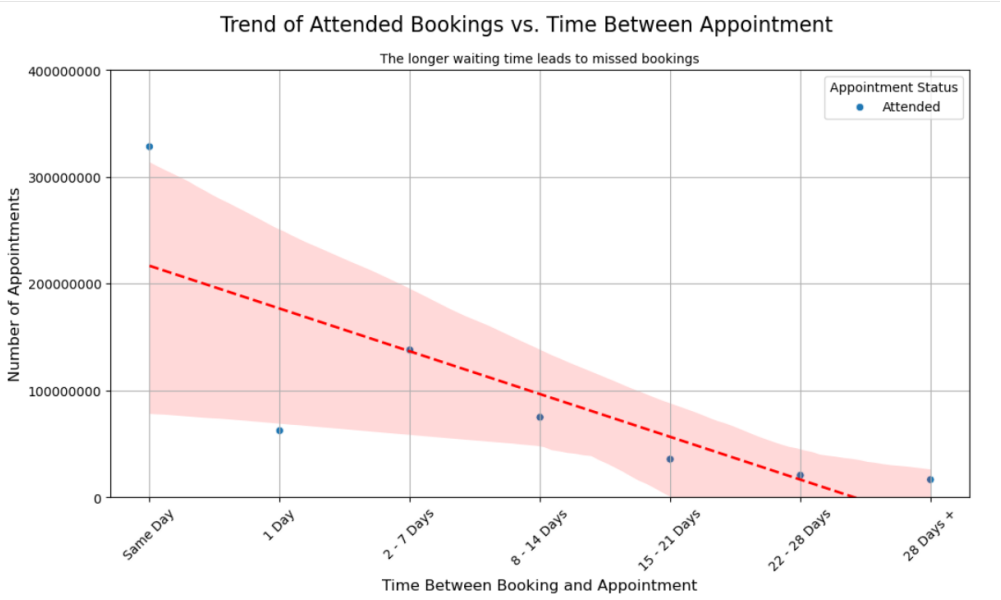


Was the waiting time affecting the appointment attendance?

To analyse attendance trends based on booking time, the data is cleaned and filtered by grouping it by the time between booking and appointment and appointment status. Irrelevant entries labelled "Unknown / Data Quality" or "Unknown" are removed, focusing on "Attended" appointments. Time categories are mapped to readable labels, reordered from "Same Day" to "More than 28 Days," and converted to numeric values for analysis. The Data Frame is then sorted to maintain the correct order.

```
# Create a table to view the trend of time between book and appointment and attended appointments

ar_df_time_clean2 = ar.groupby(['time_between_book_and_appointment','appointment_status'])['count_of_appointments'].sum().reset_index()
ar_df_time_clean2 = ar_df_time_clean2[(ar_df_time_clean2['time_between_book_and_appointment'] != "Unknown / Data Quality") &
    (ar_df_time_clean2['appointment_status'] != "Unknown")]
ar_df_time_clean2_attended = ar_df_time_clean2_attended.sort_values('count_of_appointments', ascending = False)
ar_df_time_clean2_attended=ar_df_time_clean2[ar_df_time_clean2 ['appointment_status']== 'Attended']
group_mapping = {
    'Same Day': 'Same Day',
    '1 Day': '1 Day',
    '2 to 7 Days': '2 - 7 Days',
    '8  to 14 Days': '8 - 14 Days',
    '15  to 21 Days': '15 - 21 Days',
    '22  to 28 Days': '22 - 28 Days',
    'More than 28 Days': '28 Days +'}
ar_df_time_clean2_attended['group'] = ar_df_time_clean2_attended['time_between_book_and_appointment'].map(group_mapping)

# Define the correct order
category_order = ['Same Day', '1 Day', '2 - 7 Days', '8 - 14 Days',
                  '15 - 21 Days', '22 - 28 Days', '28 Days +']
# Convert column to categorical with the correct order
ar_df_time_clean2_attended['group'] = pd.Categorical(
    ar_df_time_clean2_attended['group'],
    categories=category_order,
    ordered=True)
# Map categories to numeric values
category_map = {category: idx for idx, category in enumerate(category_order)}
ar_df_time_clean2_attended['time_numeric'] = ar_df_time_clean2_attended['group'].map(category_map)
# Sort DataFrame based on the categorical order
ar_df_time_clean2_attended = ar_df_time_clean2_attended.sort_values('group')
ar_df_time_clean2_attended
```

The analysis indicated that same-day appointments are the most popular, with patients most likely to attend when given a same-day option. There is a linear decrease in attendance as the waiting time increases.



(A trend line was added to indicate a trend of longer waiting time leads to lower appointments attended.)

Patterns and insights:

NHS GP Services Utilization & Recommendations

NHS GP service capacity has been over 75% from January 2020 to June 2022, and the trend shows this demand will continue to rise. This highlights the urgent need to hire more staff to reduce pressure on GP services.

Post-COVID, telephone consultations have grown in popularity, now making up 32% of all appointments with a 55.5% attendance rate. The number of telephone appointments grew from 3.7 million in January 2020 to 11.3 million by March 2021, with a sharp increase following the first national lockdown. The Midlands has the highest number of telephone appointments, so further investigation is needed to understand this trend.

Expanding Online GP Services

Online consultations have the highest attendance rate (66.12%), surpassing face-to-face appointments (54.53%). Given the shift in patient preferences towards virtual consultations, the NHS should focus on improving online services and developing a secure platform to make virtual consultations more accessible.

Optimizing Workforce Allocation

Appointments are busiest on weekdays, especially Mondays, which see nearly 60 million appointments. In contrast, Sundays have the lowest demand. To better meet patient needs, the NHS should allocate more staff during weekdays and especially during January-March when appointment demand peaks.

Improving Same-Day Appointment Attendance

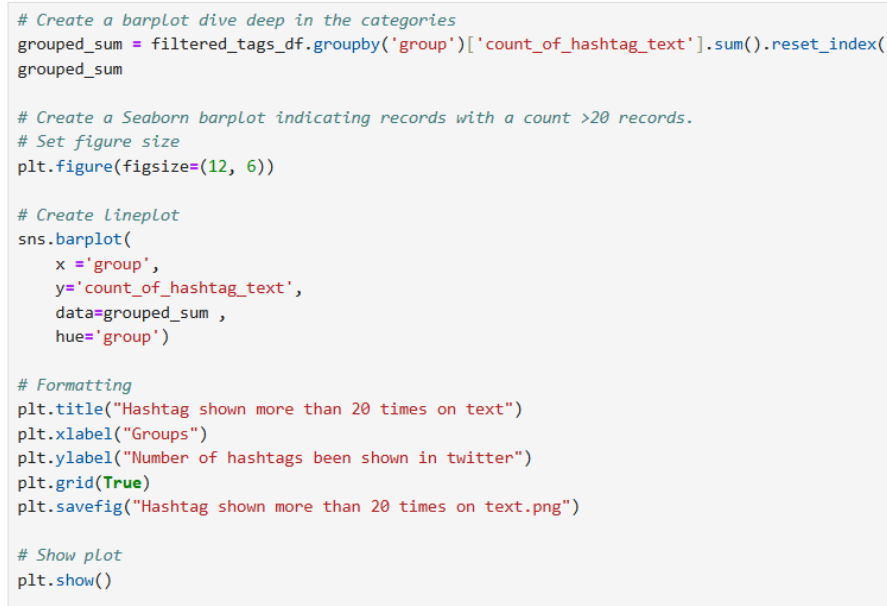
Patients prefer same-day appointments, with 48.5% choosing them. However, appointments scheduled more than 28 days in advance have a much lower attendance rate (2.5%). To improve attendance, GP practices should offer more same-day slots and send regular reminders via email or SMS to reduce no-shows.

Appendixes

Syntax	Description
`# Determine whether there are duplicate values on ar.` `ar_duplicates= ar.duplicated().sum()` `ar_duplicates` `21604`	Find the duplicates in the data frame
`# Determine whether there are duplicate values on ad.` `ad_duplicates=ad.duplicated().sum()` `ad_duplicates` `0`	Find the duplicates in the data frame
`# Determine whether there are duplicate values on nc.` `nc_duplicates=nc.duplicated().sum()` `nc_duplicates` `0`	Find the duplicates in the data frame
`# Show all duplicate rows on ar` `ar_duplicates1= ar[ar.duplicated()]` `ar_duplicates1` (table below)	Had a detail look on the duplicates confirm it's not an actual duplicate
`# Determine whether there are missing values on nc.` `nc.isnull().sum()` `appointment_date 0` `icb_ons_code 0` `sub_icb_location_name 0` `service_setting 0` `context_type 0` `national_category 0` `count_of_appointments 0` `appointment_month 0` `dtype: int64`	Confirm there is no null value in the data frame

	icb_ons_code	appointment_month	appointment_status	hcp_type	appointment_mode	time_between_book_and_appointment	count_of_appointments
19292	E54000044	2020-01	Attended	GP	Home Visit	Unknown / Data Quality	1
19308	E54000044	2020-01	Attended	GP	Unknown	Unknown / Data Quality	2
19374	E54000044	2020-01	DNA	GP	Home Visit	8 to 14 Days	1
19417	E54000044	2020-01	DNA	Unknown	Face-to-Face	2 to 7 Days	1
19419	E54000044	2020-01	DNA	Unknown	Face-to-Face	Same Day	6
...	...	...	...	...	...	...	...
596809	E54000050	2022-06	Unknown	Unknown	Telephone	15 to 21 Days	1
596812	E54000050	2022-06	Unknown	Unknown	Telephone	More than 28 Days	1

``ad.isnull().sum()``` ```sub_icb_location_code 0 sub_icb_location_ons_code 0 sub_icb_location_name 0 icb_ons_code 0 region_ons_code 0 appointment_date 0 actual_duration 0 count_of_appointments 0 dtype: int64```	Confirm there is no null value in the data frame
``# Determine whether there are missing values on ar. ar.isnull().sum()``` ```icb_ons_code 0 appointment_month 0 appointment_status 0 hcp_type 0 appointment_mode 0 time_between_book_and_appointment 0 count_of_appointments 0 dtype: int64```	
``# Plot monthly capacity utilisation. ar_df ['appointment_month'] = ar_df ['appointment_month'].astype(str) # Create a lineplot. plt.figure(figsize=(20, 10)) ax = sns.pointplot(x='appointment_month', y='utilisation_vs_capacity%', data=ar_df) sns.regplot(data=ar_df, x=ar_df.index, y='utilisation_vs_capacity%', scatter=False, color='red', label="Trend Line") # Formatting plt.title("% Capacity Utilisation in all Regions 01-2020 to 06-2022", fontsize=14) plt.xlabel(" Month", fontsize=12) plt.ylabel("Utilisation vs Capacity (%)", fontsize=12) plt.grid(True) plt.xticks(rotation=45) plt.gca().yaxis.get_major_formatter().set_useOffset(False) plt.xlim((0,30)) ax.annotate("The first national lockdown in Mar 2020 ", xy=(3, 45), xytext=(5, 45), color='red', fontsize=12, fontweight='bold', arrowprops=dict(arrowstyle="->", color='blue', linewidth=2)) plt.savefig("Utilisation per month.png") # Show plot plt.show()```	The pointplot for % Capacity Utilisa-tion in all regions from 01-2020 to 06-2022
``# Create a barplot. plt.figure(figsize=(8, 5)) sns.barplot(x='appointment_mode', y='percentage', data=status_counts, hue ='appointment_status') # Formatting plt.title("The percentage of the appointment status per appointment mode") plt.xlabel("Appointment modes") plt.ylabel("The appointments status in % ") plt.grid(True) plt.legend(title="Appointment status", bbox_to_anchor=(1,1)) plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False) # Show plot plt.show()```	The barplot for % of appointment status per appoint-ment mode

	The heatmap for viewing the telephone appointment's attendance per region
	The bar chart for twitter data to show the counts of hashtag per categories

```
# Determine the total number of appointments per month for whole year

r_df =ar.groupby(['appointment_month'])['count_of_appointments'].sum().reset_index()
# Add a new column to indicate the average utilisation of services.
# Monthly aggregate / 30 to get to a daily value.
r_df['utilisation'] = ar_df  ['count_of_appointments'] / 30

ax_capacity = 1200000
ax_capacity_rounded = round(max_capacity, 1)
r_df ['utilisation_vs_capacity%'] = ar_df ['utilisation'] / max_capacity_rounded * 100

# View the DataFrame.
r_df

# Plot monthly capacity utilisation.
r_df ['appointment_month'] = ar_df ['appointment_month'].astype(str)

# Create a lineplot.

lt.figure(figsize=(20, 10))
x = sns.pointplot(
   x='appointment_month',
   y='utilisation_vs_capacity%',
   data=ar_df)
ns.regplot(data=ar_df, x=ar_df.index, y='utilisation_vs_capacity%', scatter=False, color='red', label="Trend Lin

# Formatting
lt.title("NHS GP service is reaching over 70% utilisation during Jan 20 to Jun 22", fontsize=14)
lt.suptitle("% Capacity Utilisation in all regions from  01-2020 to 06-2022", fontsize=16)
lt.xlabel(" Month", fontsize=12)
lt.ylabel("Utilisation vs Capacity (%)", fontsize=12)
lt.grid(True)
lt.xticks(rotation=45)
lt.gca().yaxis.get_major_formatter().set_useOffset(False)
lt.xlim((0,30))
x.annotate( "The first national lockdown in Mar 2020 ",
   xy=(3, 45),
   xytext=(5, 45),
   color='red',
   fontsize=12,
   fontweight='bold',
   arrowprops=dict(arrowstyle="->", color='blue', linewidth=2))
lt.savefig("Utilisation per month.png")
```

The line chart for % Capacity Utilisation in all regions from 01-2020 to 06-2022

```
ar_clean = ar[ar['appointment_mode'] != 'Unknown']
ar_df_mode = ar_clean.groupby(['appointment_month', 'appointment_mode'])['count_of_appointments'].sum().reset_index(
ar_df_mode
# Create a line plot to answer the question.
plt.figure(figsize=(10, 6))
ax = sns.lineplot(
    x='appointment_month',
    y='count_of_appointments',
    data=ar_df_mode,
    hue = 'appointment_mode')

# Formatting
plt.xlabel("Appointment Month", fontsize=12)
plt.ylabel("Number of Appointments", fontsize=12)
plt.grid(True)
plt.xticks(rotation=45)
plt.legend(title='Appointment Modes', bbox_to_anchor=(1,1))
plt.title("How Covid-19 changed the perfered appointment mode", fontsize=10)
plt.suptitle("Trends of Appointment Mode Over Time", fontsize=16)
ax.annotate( "The first national lockdown in Mar 2020 ",
    xy=(3, 8000000),
    xytext=(5, 8000000),
    color='red',
    fontsize=12,
    fontweight='bold',
    arrowprops=dict(arrowstyle="->", color='blue', linewidth=2))
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)


# Show plot
plt.show()
```

The line chart for Trends of Appointment Mode Over Time

The line chart for Trend of Attended Bookings vs. Time Between Appointment (reference to line [899]

```
# Add a column for the day of the week
location_nc['day_of_week'] = location_nc['appointment_date'].dt.day_name()

location_nc_pivot = location_nc.pivot_table(
    values='count_of_appointments',
    index= 'day_of_week',
    aggfunc='sum').reset_index()

location_nc_pivot = location_nc_pivot.sort_values(by='count_of_appointment


# Create a bar chart to see which day has the most appointments
plt.figure(figsize=(10,6))
sns.barplot(
    y="count_of_appointments",
    x="day_of_week",
    hue="day_of_week",
    data=location_nc_pivot)

plt.title("Which day of week has more appointments?- Monday")
plt.xlabel("Day of week")
plt.ylabel("Total Appointments")
plt.grid(True)
plt.savefig("Which day of week has more appointments.png")
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)
plt.show()
```

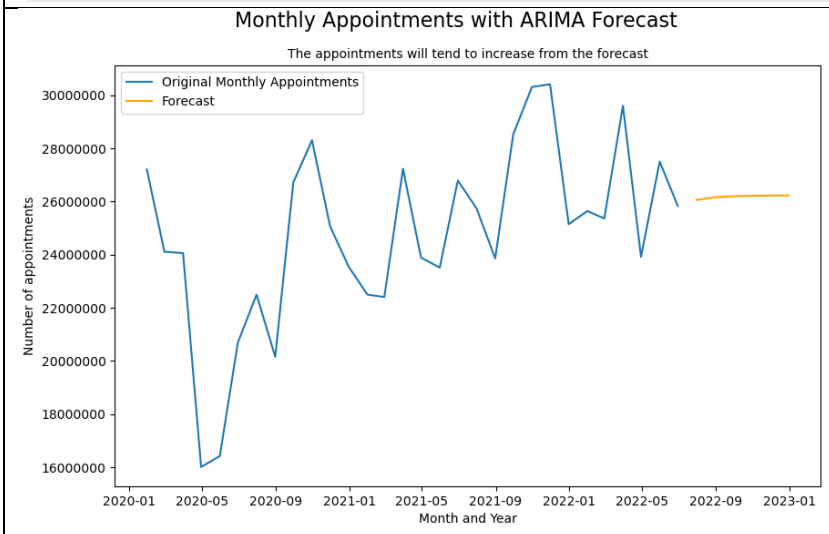
The bar chart for Which day of week has more appointments?

```
# To find which appointment mode vs appointments
# Create a ar dataframe to exclude "Unknown" in appointment mode
ar_mode_clean =  ar[ar['appointment_mode'] != "Unknown"]
ar_mode_clean

#Create a pieplot to show the appointment mode vs appointments
plt.figure(figsize=(10,6))
# Defining colors for the pie chart
colors = ['pink', 'silver', 'steelblue','blue']


plt.figure(figsize=(20, 10))

# Plotting the pie chart for above dataframe
ar_mode_clean.groupby(['appointment_mode']).count().plot(
    kind='pie', y='count_of_appointments', autopct='%1.0f%%',textprops={'fontsize': 14})


# Formatting
plt.title("The % of appointment mode vs appointments", fontsize=16)
plt.ylabel("Total Appointments", fontsize=12)
plt.legend(title='Appointment modes', bbox_to_anchor=(1,0.2))
plt.subplots_adjust(left=0.5, right=2, top=5, bottom=0.1)
```

The pie chart for The % of appointment mode vs appointments

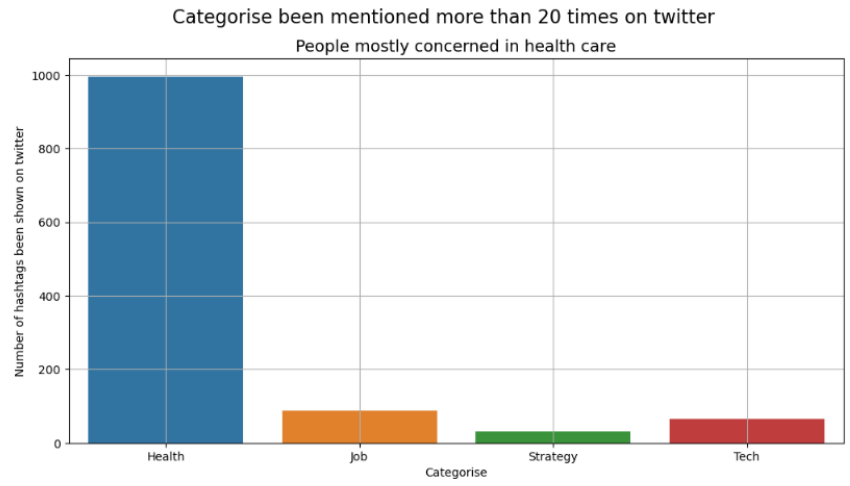
```
# Create a plot to forecast the appointments based on ar data set with ARIMA
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

ar_monthly = ar['count_of_appointments'].resample('M').sum()
ar_monthly

model = ARIMA(ar_monthly, order=(1, 1, 1))
model_fit = model.fit()

# Print summary of the model
print(model_fit.summary())

# Plot the forecasts
forecast = model_fit.forecast(steps=6)
plt.figure(figsize=(10, 6))
plt.plot(ar_monthly, label='Original Monthly Appointments')
plt.plot(forecast, label='Forecast', color='orange')
plt.title("The appointments will tend to increase from the forecast", fontsize=10)
plt.suptitle("Monthly Appointments with ARIMA Forecast", fontsize=16)
plt.xlabel('Month and Year')
plt.ylabel('Number of appointments')
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)
plt.legend()
plt.show()

forecast_values = model_fit.forecast(steps=6)
forecast_values
```

The line chart for Monthly Appoint-ments with ARIMA Forecast



```
# Create a barplot dive deep in the categories
grouped_sum = filtered_tags_df.groupby('group')['count_of_hashtag_text'].sum().reset_index()
grouped_sum

# Create a Seaborn barplot indicating records with a count >20 records.
# Set figure size
plt.figure(figsize=(12, 6))

# Create lineplot
sns.barplot(
    x ='group',
    y='count_of_hashtag_text',
    data=grouped_sum ,
    hue='group')

# Formatting
plt.title("People mostly concerned in health care", fontsize = 14)
plt.suptitle("Categorise been mentioned more than 20 times on twitter" , fontsize =16)
plt.xlabel("Categorise")
plt.ylabel("Number of hashtags been shown on twitter")
plt.grid(True)
plt.savefig("Hashtag shown more than 20 times on text.png")

# Show plot
plt.show()
```

Bar chart for twitter hastag




```
# Find the outliners of the appointments per region


location_ar['appointment_month'] = pd.to_datetime(location_ar['appointment_month'])

ar_quarter = location_ar[(location_ar['appointment_status'] == 'Attended')]

Q1_2020 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2020) & (ar_quarter['appointment_month'].dt.quarter == 1)].copy()
Q2_2020 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2020) & (ar_quarter['appointment_month'].dt.quarter == 2)].copy()
Q3_2020 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2020) & (ar_quarter['appointment_month'].dt.quarter == 3)].copy()
Q4_2020 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2020) & (ar_quarter['appointment_month'].dt.quarter == 4)].copy()

Q1_2021 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2021) & (ar_quarter['appointment_month'].dt.quarter == 1)].copy()
Q2_2021 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2021) & (ar_quarter['appointment_month'].dt.quarter == 2)].copy()
Q3_2021 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2021) & (ar_quarter['appointment_month'].dt.quarter == 3)].copy()
Q4_2021 = ar_quarter[(ar_quarter['appointment_month'].dt.year == 2021) & (ar_quarter['appointment_month'].dt.quarter == 4)].copy()

# Assign quarter labels
Q1_2020.loc[:, "Quarter"] = "Q1 2020"
Q2_2020.loc[:, "Quarter"] = "Q2 2020"
Q3_2020.loc[:, "Quarter"] = "Q3 2020"
Q4_2020.loc[:, "Quarter"] = "Q4 2020"
Q1_2021.loc[:, "Quarter"] = "Q1 2021"
Q2_2021.loc[:, "Quarter"] = "Q2 2021"
Q3_2021.loc[:, "Quarter"] = "Q3 2021"
Q4_2021.loc[:, "Quarter"] = "Q4 2021"

# Combine all quarters
combined_df = pd.concat([Q1_2020, Q2_2020, Q3_2020, Q4_2020, Q1_2021, Q2_2021, Q3_2021, Q4_2021])

# Create a box plot to see the outlier and distribution by Quaterly
plt.figure(figsize=(10, 6))
sns.boxplot(x='Quarter', y='count_of_appointments', data=combined_df)
plt.suptitle("Distribution of Attended Appointments per Quarter in 2020-2021", fontsize=16)
plt.title("Q1 (Jan to Mar) has more appointments", fontsize=14)
plt.xlabel("Quarter and Year", fontsize=12)
plt.ylabel("Total appointments", fontsize=12)
plt.show()
```

Boxplot for Quarter appointments

Quarter and Year [ ]: ``` # Found out the attendance rate ar_mode_clean = ar[ar['appointment_status'] != "Unknown"] ar_mode_clean appointment_mode_counts = ar.groupby('appointment_status')['count_of_appointments'].sum() #Create a pieplot to show the appointment mode vs appointments plt.figure(figsize=(10,6)) # Defining colors for the pie chart colors = { "Attended": "green","DNA": "red"} plt.figure(figsize=(20, 10)) # Plotting the pie chart appointment_mode_counts.plot(kind='pie', autopct='%1.0f%%', colors=[colors.get(status, 'gray') for status in appointment_mode_counts.index], textprops={'fontsize': 14}, figsize=(10, 6)) # Formatting the plot plt.title("Percentage of Appointment status- Over 90% appointments were attended", fontsize=16) plt.ylabel("") plt.legend(title='Appointment Modes', bbox_to_anchor=(1, 0.2)) plt.subplots_adjust(left=0.5, right=1.5, top=1.2, bottom=0.1) plt.show() ```	Pie chart for appointment status

Reference:

(Geographical data mapping)

https://digital.nhs.uk/data-and-information/publications/statistical/patients-registered-at-a-gp-practice/january-2025

Back up link

Back up link to Google.

https://drive.google.com/drive/folders/1w25BqEnCI9drDnUGpENapw6Re-OSbqCIj?usp=sharing