NHS Service Utilisation Analysis

Course: LSE DA201 | **Student**: Monica Baracho | **Date**: 26 May

This analysis explores the NHS's service use, missed appointments, and communication trends, providing visual and statistical evidence to inform stakeholder decision-making

This notebook requires the actual_duration.csv and appointments_regional.csv, national_categories.xlsx, icb_locations.csv, tweets.csv and metadata_nhs.txt files. Upload these files to the directory before you begin.

icb_locations.csv data extracted from ONS (Office for National Statistics) Geography portal. Population and GP (General Practitioner) data related to ICB Locations downloaded from June 2022 can be downloaded at the NHS Website.

Geographical and shape data used by Geopandas can be downloaded from the NHS website:

ICB Locations Region Locations

Navigation Tip: If you're using JupyterLab, you can open the **Table of Contents** panel from the left sidebar to navigate quickly through this notebook's sections.

In [ ]:
```
# Import necessary libraries for data handling and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load datasets
appointments = pd.read_csv('appointments_regional.csv')
duration = pd.read_csv('actual_duration.csv')
categories = pd.read_excel('national_categories.xlsx')
tweets = pd.read_csv('tweets.csv')
```

In [ ]:
```
# === Basic Dataset Overview ===

# Appointments
```

```
print("Appointments dataset:")
print(appointments.shape)
print(appointments.columns)

# Actual Duration
print("\nActual Duration dataset:")
print(duration.shape)
print(duration.columns)

# National Categories
print("\nNational Categories dataset:")
print(categories.shape)
print(categories.columns)

# Tweets
print("\nTweets dataset:")
print(tweets.shape)
print(tweets.columns)
```

Monthly Appointment Counts

This line chart visualizes the **total number of appointments per month** over time. The dataset is grouped by `appointment_month` and plotted to identify monthly trends in appointment volume.

- The **x-axis** represents time (months), starting from January 2020.
- The **y-axis** indicates the **number of appointments** (in tens of millions).
- Noticeable dips and peaks may reflect seasonal trends, policy changes, or external factors (e.g. the COVID-19 pandemic impact).

The chart helps us detect patterns in healthcare service usage and supports further investigation into possible causes behind fluctuations.

In [ ]:
```
import pandas as pd

# Load the dataset
appointments_df = pd.read_csv('appointments_regional.csv')

# Convert to datetime format if not already
appointments['appointment_month'] = pd.to_datetime(appointments['ap
categories['appointment_month'] = pd.to_datetime(categories['appoin

# Preview the data to confirm column names
appointments_df.head()
```

In [7]:
```
print(appointments['appointment_month'].dtype)
print(categories['appointment_month'].dtype)
```

```
datetime64[ns]
datetime64[ns]
```

In [8]:
```
# Group by month and sum the number of appointments
monthly_counts = appointments.groupby('appointment_month')['count_o
```

In [9]:
```
# Check all unique appointment statuses
print("Unique appointment_status values:")
print(appointments['appointment_status'].unique())
```

```
Unique appointment_status values:
['Attended' 'DNA' 'Unknown']
```

In [ ]:
```
# Load datasets first
ad = pd.read_csv('actual_duration.csv')
ar = pd.read_csv('appointments_regional.csv')
nc = pd.read_excel('national_categories.xlsx')

# Check metadata
print(ad.info())

# Check basic stats
print(ad.describe(include='all'))
```

In [ ]:
```
# Determine the descriptive statistics of the data set
# Get descriptive statistics for numeric columns
print(ad.describe())
```

In [ ]:
```

```

Missed Appointments Over Time with UK Lockdowns Highlighted

This line chart illustrates the **monthly number of missed NHS appointments (Did Not Attend - DNA)** from early 2020 to mid-2022.

- The **red line** represents the total number of missed appointments per month.
- **Shaded grey areas** correspond to periods of **UK national lockdowns** during the COVID-19 pandemic.

Key Insights:

- A **sharp drop** in missed appointments occurred in **March–April 2020**, likely due to the first national lockdown and reduced service availability.
- There was a **gradual recovery** in missed appointments during mid to late 2020, with **noticeable spikes** post-lockdowns, possibly reflecting rescheduling and increased demand.
- Lockdowns appear to have a **suppressing effect** on missed appointments, potentially due to fewer in-person visits or greater patient caution.

This visualization helps understand how public health restrictions impacted patient attendance behavior over time.

In [ ]:
```
# Load dataset
appointments = pd.read_csv("appointments_regional.csv")

# Convert to datetime
appointments['appointment_month'] = pd.to_datetime(appointments['ap

# Filter DNA (missed appointments)
missed = appointments[appointments['appointment_status'] == 'DNA']

# Group missed appointments per month
missed_by_month = missed.groupby(missed['appointment_month'].dt.to_
missed_by_month.index = missed_by_month.index.to_timestamp()

# Define key UK lockdown periods
lockdowns = [
    ('2020-03-23', '2020-07-04'),  # First lockdown
    ('2020-11-05', '2020-12-02'),  # Second lockdown
    ('2021-01-04', '2021-03-08')   # Third lockdown
]

# Plot
plt.figure(figsize=(14, 6))
plt.plot(missed_by_month.index, missed_by_month.values, marker='o',

# Add shaded regions for lockdowns
for start, end in lockdowns:
    plt.axvspan(pd.to_datetime(start), pd.to_datetime(end), color='

# Format the chart
plt.title("Missed Appointments Over Time with UK Lockdowns Highligh
plt.xlabel("Month")
plt.ylabel("Number of Missed Appointments")
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()

# Format y-axis with commas
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _:

plt.tight_layout()
plt.show()
```

Average Missed Appointments per Month

Pre-, During-, and Post-Lockdown Periods

This bar chart compares the **average number of missed NHS appointments per month** across three distinct phases of the COVID-19 pandemic:

- **Pre-Lockdown**: The highest average of missed appointments, exceeding **1.2 million per month**.
- **During Lockdown**: A significant decline in missed appointments, likely due to reduced in-person visits, cancellations, and stricter public

movement restrictions.

- **Post-Lockdown**: A rebound in missed appointments, approaching pre-lockdown levels, possibly driven by appointment backlogs and increased service demand.

Key Takeaway:

Lockdowns had a **clear suppressing effect** on missed appointments, but this effect was temporary. The post-lockdown recovery indicates a return to typical patient behavior and NHS capacity levels.

This analysis provides insight into how external policy measures (e.g. lockdowns) can influence patient engagement and service delivery trends.

In [ ]:
```
# Load the dataset
appointments = pd.read_csv("appointments_regional.csv")

# Convert to datetime
appointments['appointment_month'] = pd.to_datetime(appointments['ap

# Filter for missed appointments (DNA)
missed = appointments[appointments['appointment_status'] == 'DNA']

# Define the three key periods
pre_lockdown = missed[missed['appointment_month'] < '2020-03-23']
lockdown = missed[(missed['appointment_month'] >= '2020-03-23') & (
post_lockdown = missed[missed['appointment_month'] > '2021-03-08']

# Group by month and calculate the sum for each period
pre_summary = pre_lockdown.groupby(pre_lockdown['appointment_month'
lockdown_summary = lockdown.groupby(lockdown['appointment_month'].d
post_summary = post_lockdown.groupby(post_lockdown['appointment_mon

# Calculate average number of missed appointments per month
averages = pd.Series({
    'Pre-Lockdown': pre_summary.mean(),
    'Lockdown': lockdown_summary.mean(),
    'Post-Lockdown': post_summary.mean()
})

# Plot the results
plt.figure(figsize=(8, 5))
averages.plot(kind='bar', color=['steelblue', 'gray', 'seagreen'])

# Formatting
plt.title("Average Missed Appointments per Month\nPre-, During-, an
plt.ylabel("Average Monthly Missed Appointments")
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, _:
plt.tight_layout()
plt.show()
```

In [16]:
```
# Get descriptive statistics for all columns, including object type
print("\n Descriptive statistics (including object columns):\n")
print(ad.describe(include='all'))
```

```
 Descriptive statistics (including object columns):

       sub_icb_location_code sub_icb_location_ons_code  \
count                 137793                    137793
unique                   106                       106
top                      26A                 E38000239
freq                    1484                      1484
mean                     NaN                       NaN
std                      NaN                       NaN
min                      NaN                       NaN
25%                      NaN                       NaN
50%                      NaN                       NaN
75%                      NaN                       NaN
max                      NaN                       NaN

                   sub_icb_location_name icb_ons_code region_ons_co
de  \
count                             137793       137793          1377
93
unique                               106           42
7
top     NHS Norfolk and Waveney ICB - 26A    E54000057       E400000
10
freq                                1484        12668           331
12
mean                                 NaN          NaN             N
aN
std                                  NaN          NaN             N
aN
min                                  NaN          NaN             N
aN
25%                                  NaN          NaN             N
aN
50%                                  NaN          NaN             N
aN
75%                                  NaN          NaN             N
aN
max                                  NaN          NaN             N
aN

       appointment_date         actual_duration  count_of_appointmen
ts
count            137793                  137793          137793.0000
00
unique              212                       7                   N
aN
top           01-Dec-21  Unknown / Data Quality                   N
aN
freq                742                   20161                   N
aN
mean                NaN                     NaN           1219.0800
11
```

```
std                  NaN                  NaN            1546.9029
56
min                  NaN                  NaN               1.0000
00
25%                  NaN                  NaN             194.0000
00
50%                  NaN                  NaN             696.0000
00
75%                  NaN                  NaN            1621.0000
00
max                  NaN                  NaN           15400.0000
00
```

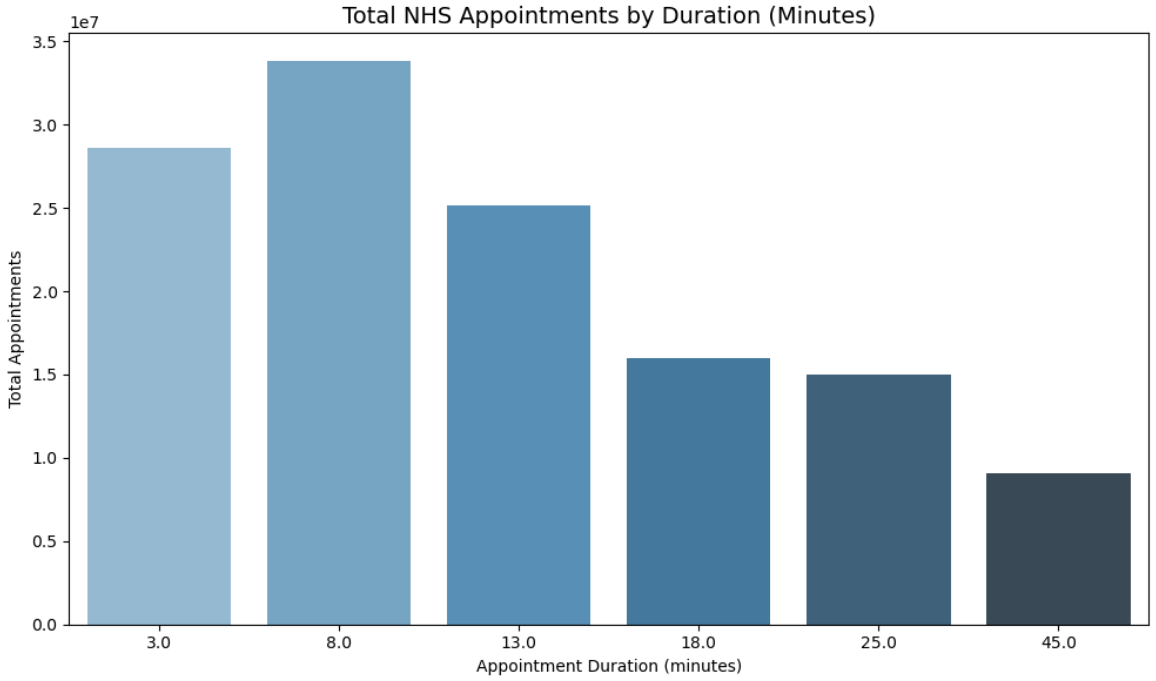
In [17]:
```
# Check unique values and counts in actual_duration
print(ad['actual_duration'].value_counts())
```

```
actual_duration
Unknown / Data Quality    20161
1-5 Minutes               19909
6-10 Minutes              19902
11-15 Minutes             19738
16-20 Minutes             19534
21-30 Minutes             19452
31-60 Minutes             19097
Name: count, dtype: int64
```

In [18]:
```
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Step 2: Load the actual_duration dataset (if not already loaded)
ad = pd.read_csv('actual_duration.csv')

# Step 3: Map duration categories to midpoint values (in minutes)
duration_map = {
    '1-5 Minutes': 3,
    '6-10 Minutes': 8,
    '11-15 Minutes': 13,
    '16-20 Minutes': 18,
    '21-30 Minutes': 25,
    '31-60 Minutes': 45,
    'Unknown / Data Quality': np.nan
}

# Step 4: Create a numeric column for duration
ad['duration_minutes'] = ad['actual_duration'].map(duration_map)

# Step 5: Drop rows with unknown duration
duration_clean = ad.dropna(subset=['duration_minutes'])

# Step 6: Plot total appointments by duration
plt.figure(figsize=(10, 6))
sns.barplot(
    data=duration_clean,
    x='duration_minutes',
```

```
    y='count_of_appointments',
    estimator=sum,
    ci=None,
    palette='Blues_d'
)

# Step 7: Format and show plot
plt.title('Total NHS Appointments by Duration (Minutes)', fontsize=
plt.xlabel('Appointment Duration (minutes)')
plt.ylabel('Total Appointments')
plt.tight_layout()
plt.show()
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/387
3081946.py:29: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same e
ffect.

  sns.barplot(
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/387
3081946.py:29: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(
```



In [19]:
```
# Import and sense-check the appointments_regional.csv data set as a
ar = pd.read_csv('appointments_regional.csv')

# View the DataFrame.
print(ar.info())
print(ar.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 596821 entries, 0 to 596820
Data columns (total 7 columns):
 #   Column                            Non-Null Count   Dtype 
---  ------                            --------------   ----- 
 0   icb_ons_code                      596821 non-null  object
 1   appointment_month                 596821 non-null  object
 2   appointment_status                596821 non-null  object
 3   hcp_type                          596821 non-null  object
 4   appointment_mode                  596821 non-null  object
 5   time_between_book_and_appointment  596821 non-null  object
 6   count_of_appointments             596821 non-null  int64 
dtypes: int64(1), object(6)
memory usage: 31.9+ MB
None
  icb_ons_code appointment_month appointment_status hcp_type appoint
ment_mode  \
0    E54000034           2020-01           Attended       GP     Fac
e-to-Face
1    E54000034           2020-01           Attended       GP     Fac
e-to-Face
2    E54000034           2020-01           Attended       GP     Fac
e-to-Face
3    E54000034           2020-01           Attended       GP     Fac
e-to-Face
4    E54000034           2020-01           Attended       GP     Fac
e-to-Face

  time_between_book_and_appointment  count_of_appointments
0                            1 Day                  8107
1                  15  to 21 Days                  6791
2                     2 to 7 Days                 20686
3                  22  to 28 Days                  4268
4                   8  to 14 Days                 11971
```

In [20]:
```
# Convert appointment_month to string (YYYY-MM) or datetime
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'])
ar['appointment_month_str'] = ar['appointment_month'].dt.strftime('
```

In [21]:
```
# Check unique appointment statuses and modes
print(ar['appointment_status'].unique())
print(ar['appointment_mode'].unique())
```

```
['Attended' 'DNA' 'Unknown']
['Face-to-Face' 'Home Visit' 'Telephone' 'Unknown' 'Video/Online']
```

In [22]:
```
# Determine whether there are missing values.
missing_values_ar = ar.isnull().sum()
print("Missing values per column in appointments_regional:\n")
print(missing_values_ar)
```

```
Missing values per column in appointments_regional:

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
appointment_month_str                0
dtype: int64
```

Missing Values Check — appointments_regional.csv

We checked for missing values in the `appointments_regional.csv` dataset.

Result:

- No missing values in any of the 8 columns.
- Columns like `appointment_status` and `appointment_mode` contain the category `'Unknown'`, but this is a data value — not a missing entry.

No imputation or removal of missing data is required.

In [24]:
```
# Determine the metadata of the data set.
print(ar.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 596821 entries, 0 to 596820
Data columns (total 8 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   icb_ons_code                        596821 non-null  object
 1   appointment_month                   596821 non-null  datetime64[
ns]
 2   appointment_status                  596821 non-null  object
 3   hcp_type                            596821 non-null  object
 4   appointment_mode                    596821 non-null  object
 5   time_between_book_and_appointment   596821 non-null  object
 6   count_of_appointments               596821 non-null  int64
 7   appointment_month_str               596821 non-null  object
dtypes: datetime64[ns](1), int64(1), object(6)
memory usage: 36.4+ MB
None
```

In [25]:
```
# Determine the descriptive statistics of the data set.
# Get descriptive statistics for numeric columns
print(ar.describe())

# Get descriptive statistics for all columns, including object type
print("\n Descriptive statistics (including object columns):\n")
```

```
print(ar.describe(include='all'))
```

```
                 appointment_month  count_of_appointments
count                       596821          596821.000000
mean   2021-03-19 11:31:17.489565696            1244.601857
min              2020-01-01 00:00:00               1.000000
25%              2020-08-01 00:00:00               7.000000
50%              2021-04-01 00:00:00              47.000000
75%              2021-11-01 00:00:00             308.000000
max              2022-06-01 00:00:00          211265.000000
std                              NaN            5856.887042

 Descriptive statistics (including object columns):

       icb_ons_code              appointment_month appointment_statu
s  \
count        596821                         596821             59682
1
unique           42                            NaN
3
top       E54000057                            NaN           Attende
d
freq          54527                            NaN             23213
7
mean            NaN  2021-03-19 11:31:17.489565696                Na
N
min             NaN            2020-01-01 00:00:00                Na
N
25%             NaN            2020-08-01 00:00:00                Na
N
50%             NaN            2021-04-01 00:00:00                Na
N
75%             NaN            2021-11-01 00:00:00                Na
N
max             NaN            2022-06-01 00:00:00                Na
N
std             NaN                            NaN                Na
N


                   hcp_type appointment_mode  \
count                596821           596821
unique                    3                5
top     Other Practice staff     Face-to-Face
freq                 241557           180410
mean                    NaN              NaN
min                     NaN              NaN
25%                     NaN              NaN
50%                     NaN              NaN
75%                     NaN              NaN
max                     NaN              NaN
std                     NaN              NaN

       time_between_book_and_appointment  count_of_appointments  \
count                             596821          596821.000000
unique                                 8                    NaN
top                             Same Day                    NaN
```

```
freq                                95502                  NaN
mean                                  NaN          1244.601857
min                                   NaN             1.000000
25%                                   NaN             7.000000
50%                                   NaN            47.000000
75%                                   NaN           308.000000
max                                   NaN        211265.000000
std                                   NaN          5856.887042

       appointment_month_str
count                 596821
unique                    30
top                  2020-03
freq                   21350
mean                     NaN
min                      NaN
25%                      NaN
50%                      NaN
75%                      NaN
max                      NaN
std                      NaN
```

In [ ]:
```
# Import pandas if not already imported
import pandas as pd

# Import and sense-check the national_categories.xlsx data set as nc
nc = pd.read_excel('national_categories.xlsx')

# View the structure and first few rows of the DataFrame
print(nc.info())
print(nc.head())
```

In [ ]:
```
# Determine whether there are missing values.
# Check for missing values in the nc DataFrame
missing_values_nc = nc.isnull().sum()
print("Missing values per column in national_categories:\n")
print(missing_values_nc)
```

In [ ]:
```
# Determine the metadata of the data set.
# Check the metadata of the nc DataFrame
print(nc.info())
```

Activity 2: Importing and exploring the data

Descriptive Statistics — national_categories.xlsx

We used `.describe()` to explore the dataset.

Numeric column: `count_of_appointments`

- Mean: ~362 appointments
- Min: 1 appointment
- Max: 16,590 appointments (large outlier)
- Standard deviation: ~1,085 (high variability)
- Median (50%): 25 appointments
- 25th percentile: 7 appointments
- 75th percentile: 128 appointments

Insight:
There is a highly uneven distribution of appointments, with some records showing very large counts. This suggests the presence of outliers and aggregation at different levels.

Date column: `appointment_date`

- Range: August 1, 2021 → June 30, 2022
- Median date: January 18, 2022

Categorical columns

- `icb_ons_code`: 42 unique codes; top = `E54000057` (59,180 rows)
- `sub_icb_location_name`: 106 unique locations; top = `NHS North West London ICB – W2U3Z` (13,007 rows)
- `service_setting`: 5 categories; top = `General Practice` (359,274 rows)
- `context_type`: 3 categories; top = `Care Related Encounter` (700,481 rows)
- `national_category`: 18 categories; top = `Inconsistent Mapping` (89,494 rows)
- `appointment_month`: 11 months; top = March 2022 (82,822 rows)

Key takeaways

- The dataset covers 11 months with heavy representation in March 2022.
- `General Practice` and `Care Related Encounter` dominate service settings and context types.
- `Inconsistent Mapping` as the top `national_category` may require cleaning or review.

Explore the data set

Question 1: How many locations are there in the data set?

In [32]:
```
# Step 1: Load and inspect files
region_meta = pd.read_csv("NHS_England_Regions_July_2022_EN_BFC.csv'
icb_lookup = pd.read_csv("icb_locations.csv")

# Step 2: Merge directly using available names
merged_icb = icb_lookup.merge(
    region_meta[['NHSER22CD', 'NHSER22NM']],
    left_on='region_ons_code', right_on='NHSER22CD',
    how='left'
)

# Step 3: Merge region info into national_categories data
nc_merged = nc.merge(merged_icb[['icb_ons_code', 'NHSER22NM']], on=

# Step 4: Count entities
print(f"Sub-ICB Locations = {nc_merged['sub_icb_location_name'].nun
print(f"ICBs = {nc_merged['icb_ons_code'].nunique()}")
print(f"Regions = {nc_merged['NHSER22NM'].nunique()}")
```

```
Sub-ICB Locations = 106
ICBs = 42
Regions = 7
```

In [33]:
```
# Optional: list all unique location names
unique_locations = nc['sub_icb_location_name'].unique()
print("Unique location names:\n")
print(unique_locations)
```

```
Unique location names:

['NHS North East and North Cumbria ICB - 00L'
 'NHS North East and North Cumbria ICB - 00N'
 'NHS North East and North Cumbria ICB - 00P'
 'NHS Lancashire and South Cumbria ICB - 00Q'
 'NHS Lancashire and South Cumbria ICB - 00R'
 'NHS Greater Manchester ICB - 00T' 'NHS Greater Manchester ICB - 00
V'
 'NHS Lancashire and South Cumbria ICB - 00X'
 'NHS Greater Manchester ICB - 00Y'
 'NHS Lancashire and South Cumbria ICB - 01A'
 'NHS Greater Manchester ICB - 01D'
 'NHS Lancashire and South Cumbria ICB - 01E'
 'NHS Cheshire and Merseyside ICB - 01F'
 'NHS Greater Manchester ICB - 01G'
 'NHS North East and North Cumbria ICB - 01H'
 'NHS Cheshire and Merseyside ICB - 01J'
 'NHS Lancashire and South Cumbria ICB - 01K'
 'NHS Cheshire and Merseyside ICB - 01T'
 'NHS Cheshire and Merseyside ICB - 01V'
 'NHS Greater Manchester ICB - 01W'
 'NHS Cheshire and Merseyside ICB - 01X'
 'NHS Greater Manchester ICB - 01Y' 'NHS Greater Manchester ICB - 02
A'
 'NHS Cheshire and Merseyside ICB - 02E'
 'NHS Lancashire and South Cumbria ICB - 02G'
 'NHS Greater Manchester ICB - 02H'
```

```
 'NHS Lancashire and South Cumbria ICB - 02M'
 'NHS South Yorkshire ICB - 02P'
 'NHS Nottingham and Nottinghamshire ICB - 02Q'
 'NHS West Yorkshire ICB - 02T' 'NHS South Yorkshire ICB - 02X'
 'NHS Humber and North Yorkshire ICB - 02Y'
 'NHS Humber and North Yorkshire ICB - 03F'
 'NHS Humber and North Yorkshire ICB - 03H'
 'NHS Humber and North Yorkshire ICB - 03K'
 'NHS South Yorkshire ICB - 03L' 'NHS South Yorkshire ICB - 03N'
 'NHS Humber and North Yorkshire ICB - 03Q' 'NHS West Yorkshire ICB
- 03R'
 'NHS Leicester Leicestershire and Rutland ICB - 03W'
 'NHS Leicester Leicestershire and Rutland ICB - 04C'
 'NHS Leicester Leicestershire and Rutland ICB - 04V'
 'NHS Staffordshire and Stoke-on-Trent ICB - 04Y'
 'NHS Staffordshire and Stoke-on-Trent ICB - 05D'
 'NHS Staffordshire and Stoke-on-Trent ICB - 05G'
 'NHS Staffordshire and Stoke-on-Trent ICB - 05Q'
 'NHS Staffordshire and Stoke-on-Trent ICB - 05V'
 'NHS Staffordshire and Stoke-on-Trent ICB - 05W'
 'NHS Cambridgeshire and Peterborough ICB - 06H'
 'NHS Hertfordshire and West Essex ICB - 06K'
 'NHS Suffolk and North East Essex ICB - 06L'
 'NHS Hertfordshire and West Essex ICB - 06N'
 'NHS Mid and South Essex ICB - 06Q'
 'NHS Suffolk and North East Essex ICB - 06T'
 'NHS Mid and South Essex ICB - 07G'
 'NHS Hertfordshire and West Essex ICB - 07H'
 'NHS Suffolk and North East Essex ICB - 07K' 'NHS Sussex ICB - 09D'
 'NHS Buckinghamshire Oxfordshire and Berkshire West ICB - 10Q'
 'NHS Hampshire and Isle Of Wight ICB - 10R' 'NHS Dorset ICB - 11J'
 'NHS Gloucestershire ICB - 11M'
 'NHS Cornwall and The Isles Of Scilly ICB - 11N' 'NHS Somerset ICB
- 11X'
 'NHS Cheshire and Merseyside ICB - 12F'
 'NHS North East and North Cumbria ICB - 13T'
 'NHS Greater Manchester ICB - 14L'
 'NHS Buckinghamshire Oxfordshire and Berkshire West ICB - 14Y'
 'NHS Buckinghamshire Oxfordshire and Berkshire West ICB - 15A'
 'NHS Bristol North Somerset and South Gloucestershire ICB - 15C'
 'NHS Birmingham and Solihull ICB - 15E' 'NHS West Yorkshire ICB - 1
5F'
 'NHS Derby and Derbyshire ICB - 15M' 'NHS Devon ICB - 15N'
 'NHS North East and North Cumbria ICB - 16C'
 'NHS Herefordshire and Worcestershire ICB - 18C'
 'NHS Norfolk and Waveney ICB - 26A'
 'NHS Cheshire and Merseyside ICB - 27D' 'NHS West Yorkshire ICB - 3
6J'
 'NHS South West London ICB - 36L'
 'NHS Humber and North Yorkshire ICB - 42D'
 'NHS Nottingham and Nottinghamshire ICB - 52R' 'NHS Sussex ICB - 70
F'
 'NHS Lincolnshire ICB - 71E' 'NHS South East London ICB - 72Q'
 'NHS Northamptonshire ICB - 78H'
 'NHS North East and North Cumbria ICB - 84H'
 'NHS Kent and Medway ICB - 91Q' 'NHS Surrey Heartlands ICB - 92A'
```

```
 'NHS Bath and North East Somerset Swindon and Wiltshire ICB - 92G'
 'NHS North Central London ICB - 93C' 'NHS Sussex ICB - 97R'
 'NHS Cheshire and Merseyside ICB - 99A'
 'NHS North East and North Cumbria ICB - 99C'
 'NHS Mid and South Essex ICB - 99E' 'NHS Mid and South Essex ICB -
99F'
 'NHS Mid and South Essex ICB - 99G' 'NHS North East London ICB - A3
A8R'
 'NHS Coventry and Warwickshire ICB - B2M3M'
 'NHS Black Country ICB - D2P2L' 'NHS Frimley ICB - D4U1Y'
 'NHS Hampshire and Isle Of Wight ICB - D9Y0V'
 'NHS Bedfordshire Luton and Milton Keynes ICB - M1J4Y'
 'NHS Shropshire Telford and Wrekin ICB - M2L0M'
 'NHS North West London ICB - W2U3Z' 'NHS West Yorkshire ICB - X2C4
Y']
```

Question 2: What are the five locations with the highest number of appointments?

This was calculated as the sum of count_of_appointments by sub_icb_location_name from the dataset national_categories.xlsx

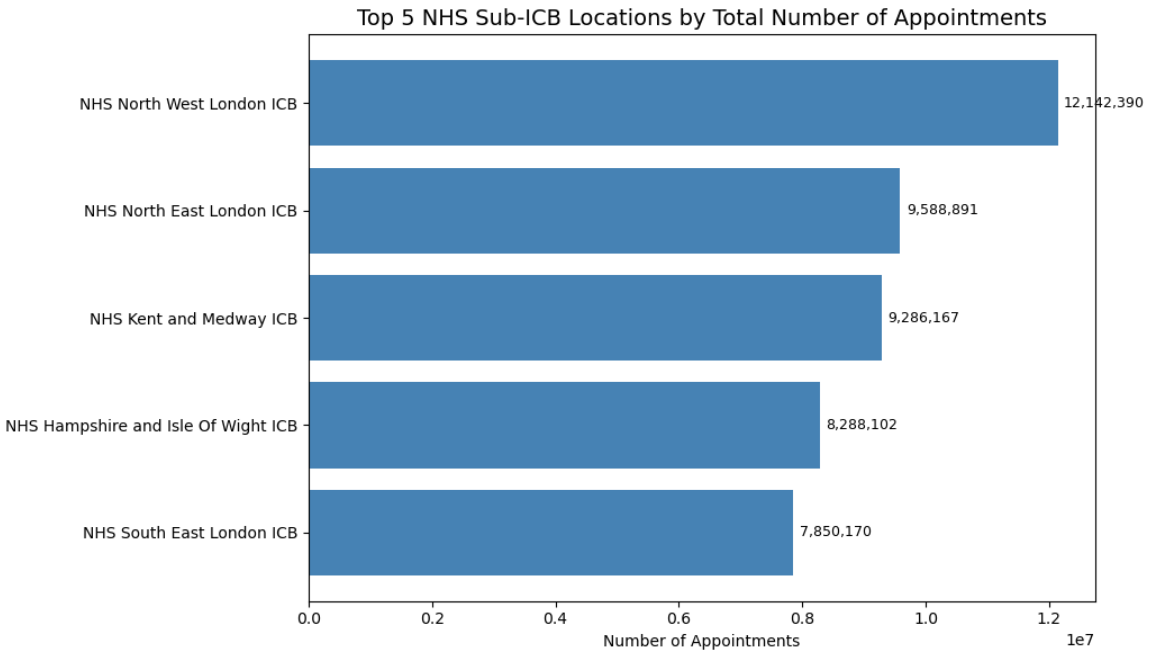
In [35]:
```
# Determine the top five locations based on record count
# Group by location and sum the number of appointments
top_locations = nc.groupby('sub_icb_location_name')['count_of_appoi

# Sort in descending order and get the top
top_five_locations = top_locations.sort_values(ascending=False).hea

# Display the result
print("Top five locations by total number of appointments:\n")
print(top_five_locations)
```

```
Top five locations by total number of appointments:

sub_icb_location_name
NHS North West London ICB - W2U3Z              12142390
NHS North East London ICB - A3A8R               9588891
NHS Kent and Medway ICB - 91Q                   9286167
NHS Hampshire and Isle Of Wight ICB - D9Y0V     8288102
NHS South East London ICB - 72Q                 7850170
Name: count_of_appointments, dtype: int64
```

In [36]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Example top 5 data (from your previous analysis)
top_five_data = {
    'sub_icb_location_name': [
        'NHS North West London ICB',
        'NHS North East London ICB',
        'NHS Kent and Medway ICB',
        'NHS Hampshire and Isle Of Wight ICB',
        'NHS South East London ICB'
    ],
    'count_of_appointments': [
```

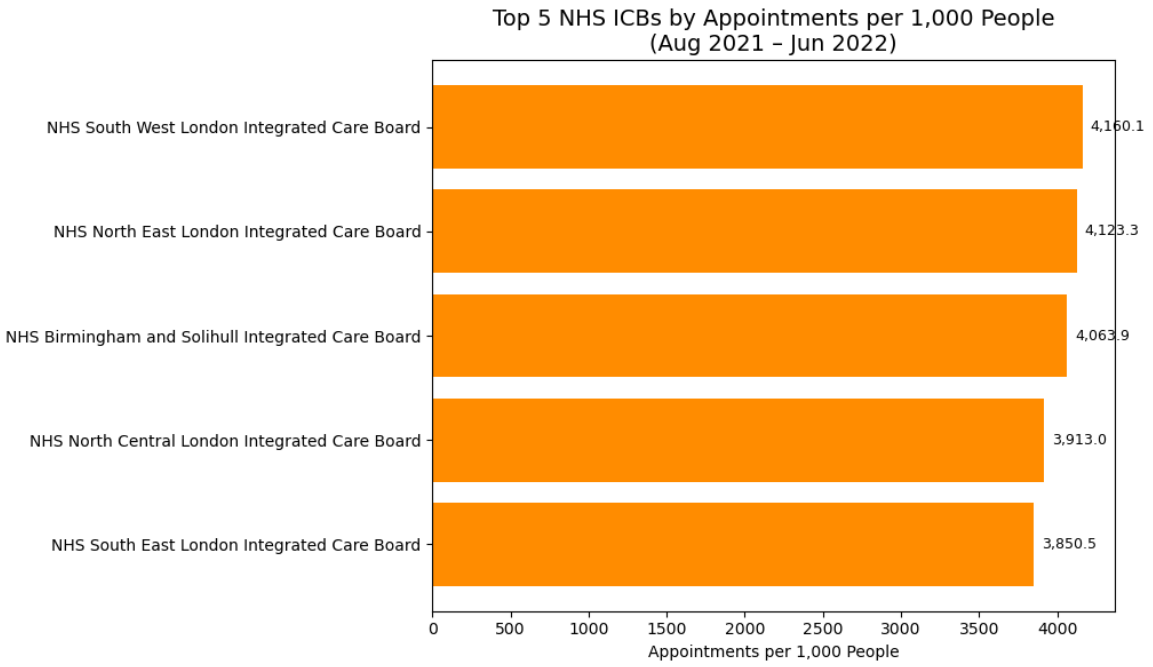
```
        12142390,
        9588891,
        9286167,
        8288102,
        7850170
    ]
}

# Convert to DataFrame
df = pd.DataFrame(top_five_data)


# Sort for nicer plotting (optional)
df = df.sort_values('count_of_appointments', ascending=True)

# Create figure
plt.figure(figsize=(10, 6))
bars = plt.barh(df['sub_icb_location_name'], df['count_of_appointme

# Add title and labels
plt.title('Top 5 NHS Sub-ICB Locations by Total Number of Appointme
plt.xlabel('Number of Appointments')
plt.tight_layout()

# Add value labels on the bars
for bar in bars:
    width = bar.get_width()
    plt.text(width + 100000, bar.get_y() + bar.get_height()/2,
             f'{width:,}', va='center', fontsize=9)

# Save the figure
plt.savefig('top_five_appointments_barchart.png', dpi=300, bbox_incl

print("Bar chart saved as 'top_five_appointments_barchart.png'")
```

```
Bar chart saved as 'top_five_appointments_barchart.png'
```



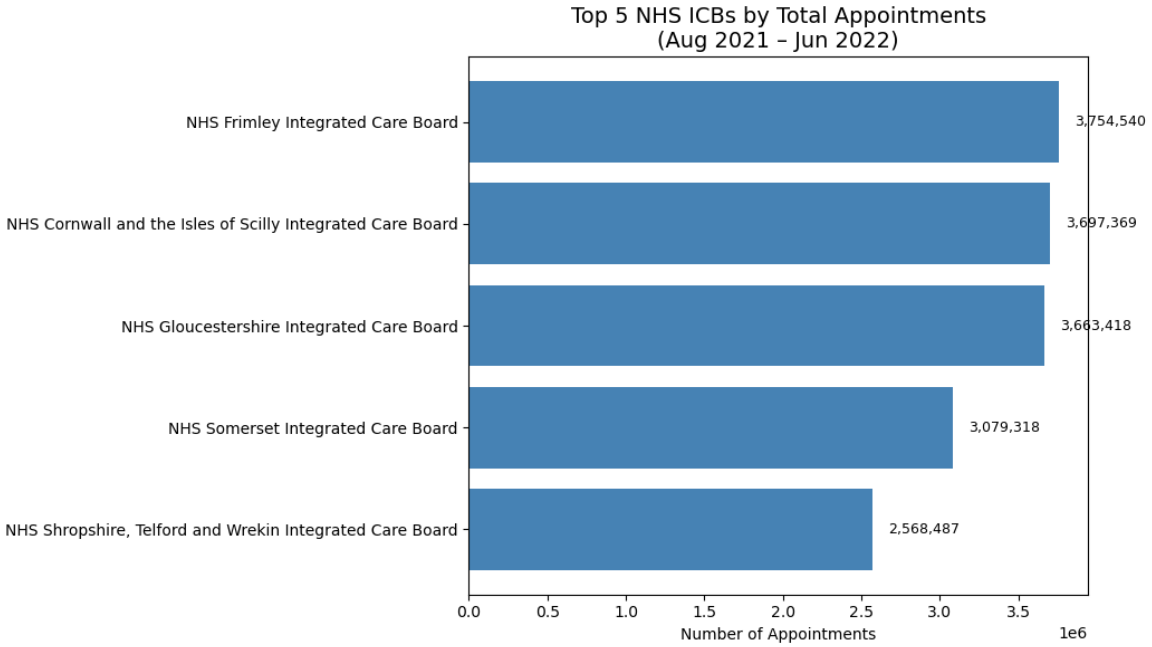
In [37]:
```
import pandas as pd
```

```
import matplotlib.pyplot as plt

# Load datasets
appointments_df = pd.read_csv("appointments_regional.csv", parse_da
icb_df = pd.read_csv("icb_locations.csv")

# Filter to the correct date range
appointments_filtered = appointments_df[
    (appointments_df["appointment_month"] >= "2021-08-01") &
    (appointments_df["appointment_month"] <= "2022-06-30")
]

# Total appointments by ICB
total_appointments = appointments_filtered.groupby("icb_ons_code")['

# Merge with population data
merged = total_appointments.merge(
    icb_df[["icb_ons_code", "icb_ons_name", "icb_registered_populat
    on="icb_ons_code",
    how="left"
)

# Calculate appointments per 1,000 people
merged["appointments_per_1000"] = (
    merged["count_of_appointments"] / merged["icb_registered_popula
) * 1000

# Sort and take top 5
top5 = merged.sort_values(by="appointments_per_1000", ascending=Tru

# Plot
plt.figure(figsize=(10, 6))
bars = plt.barh(top5["icb_ons_name"], top5["appointments_per_1000"]
plt.title("Top 5 NHS ICBs by Appointments per 1,000 People\n(Aug 20
plt.xlabel("Appointments per 1,000 People")
plt.tight_layout()

# Add value labels
for bar in bars:
    width = bar.get_width()
    plt.text(width + 50, bar.get_y() + bar.get_height() / 2, f'{wid

# Save high-res image
plt.savefig("Top5_ICBs_Appointments_Per_1000.png", dpi=300, bbox_in

# Show plot
plt.show()
```



In [38]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data
appointments_df = pd.read_csv("appointments_regional.csv", parse_da
icb_df = pd.read_csv("icb_locations.csv")

# Filter data for August 2021 – June 2022
appointments_filtered = appointments_df[
    (appointments_df["appointment_month"] >= "2021-08-01") &
    (appointments_df["appointment_month"] <= "2022-06-30")
]

# Aggregate total appointments per ICB
total_per_icb = (
    appointments_filtered.groupby("icb_ons_code")["count_of_appointr
    .sum()
    .reset_index()
)

# Merge with ICB names
merged = total_per_icb.merge(
    icb_df[["icb_ons_code", "icb_ons_name"]],
    on="icb_ons_code",
    how="left"
)

# Get top 5 by appointment volume
top5 = merged.sort_values(by="count_of_appointments", ascending=Tru

# Plot
plt.figure(figsize=(10, 6))
bars = plt.barh(top5["icb_ons_name"], top5["count_of_appointments"]
plt.title("Top 5 NHS ICBs by Total Appointments\n(Aug 2021 – Jun 20
plt.xlabel("Number of Appointments")
plt.tight_layout()
```

```
# Add value labels to bars
for bar in bars:
    width = bar.get_width()
    plt.text(width + 100000, bar.get_y() + bar.get_height() / 2,
             f'{width:,}', va='center', fontsize=9)

# Save high-resolution PNG
plt.savefig("Top5_ICBs_Total_Appointments.png", dpi=300, bbox_inche

# Show chart
plt.show()

# Display data for reference
top5[["icb_ons_name", "count_of_appointments"]]
```



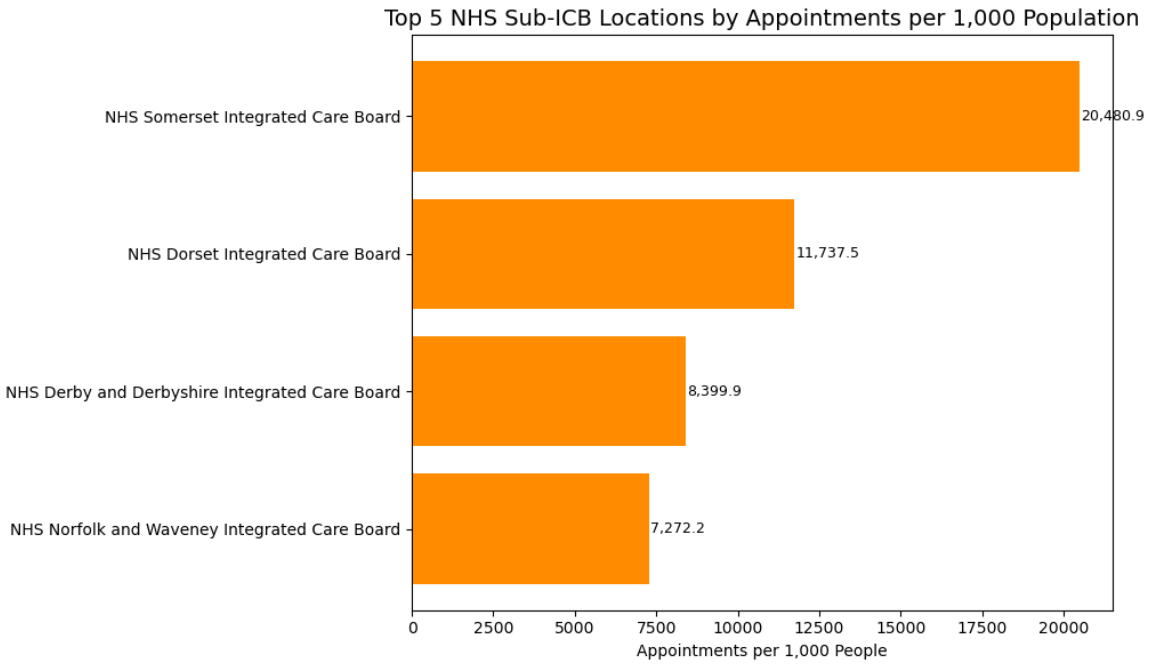
Out[38]:

	icb_ons_name	count_of_appointments
2	NHS Shropshire, Telford and Wrekin Integrated ...	2568487
21	NHS Somerset Integrated Care Board	3079318
26	NHS Gloucestershire Integrated Care Board	3663418
19	NHS Cornwall and the Isles of Scilly Integrate...	3697369
18	NHS Frimley Integrated Care Board	3754540

In [39]:

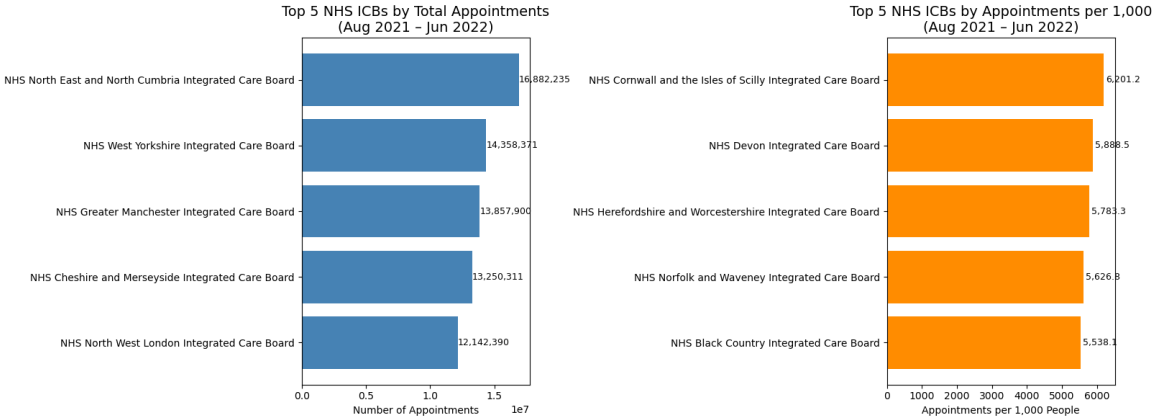
```
import pandas as pd
import matplotlib.pyplot as plt

# Load your population data
icb_df = pd.read_csv('icb_locations.csv')

# Define correct ICB codes and their total appointments
top5_mapping = {
    'E54000038': 'NHS North West London ICB',
    'E54000041': 'NHS North East London ICB',
```

```
    'E54000058': 'NHS Kent and Medway ICB',
    'E54000016': 'NHS Hampshire and Isle Of Wight ICB',
    'E54000022': 'NHS South East London ICB'
}

appointment_counts = {
    'E54000038': 12142390,
    'E54000041': 9588891,
    'E54000058': 9286167,
    'E54000016': 8288102,
    'E54000022': 7850170
}

# Filter top 5 ICB data
top5_icb_data = icb_df[icb_df['icb_ons_code'].isin(top5_mapping.key
    ['icb_ons_code', 'icb_ons_name', 'icb_registered_population_202
]

# Add total appointments
top5_icb_data['count_of_appointments'] = top5_icb_data['icb_ons_cod

# Calculate appointments per 1,000 population
top5_icb_data['appointments_per_1000'] = (
    top5_icb_data['count_of_appointments'] / top5_icb_data['icb_reg
) * 1000

# Sort for better plotting
plot_df = top5_icb_data.sort_values('appointments_per_1000', ascend

# Plot horizontal bar chart
plt.figure(figsize=(10, 6))
bars = plt.barh(plot_df['icb_ons_name'], plot_df['appointments_per_

# Add title and labels
plt.title('Top 5 NHS Sub-ICB Locations by Appointments per 1,000 Po
plt.xlabel('Appointments per 1,000 People')
plt.tight_layout()

# Add value labels on bars
for bar in bars:
    width = bar.get_width()
    plt.text(width + 50, bar.get_y() + bar.get_height() / 2,
             f'{width:,.1f}', va='center', fontsize=9)

# Save as high-resolution PNG
plt.savefig('top_five_appointments_per_1000_barchart.png', dpi=300,

# Show plot in notebook
plt.show()
```



In [40]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load datasets
appointments = pd.read_csv('appointments_regional.csv', parse_dates
icb_lookup = pd.read_csv('icb_locations.csv')

# Filter by date range
appointments_filtered = appointments[
    (appointments['appointment_month'] >= '2021-08-01') &
    (appointments['appointment_month'] <= '2022-06-30')
]

# Total appointments by ICB
icb_totals = appointments_filtered.groupby('icb_ons_code')['count_o

# Merge with ICB names and population
merged = icb_totals.merge(icb_lookup[['icb_ons_code', 'icb_ons_name
                          on='icb_ons_code')

# Compute per 1,000 population
merged['appointments_per_1000'] = (
    merged['count_of_appointments'] / merged['icb_registered_popula
) * 1000

# Sort and select top 5
top_total = merged.sort_values('count_of_appointments', ascending=F
top_1000 = merged.sort_values('appointments_per_1000', ascending=Fa

# Plot
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Total appointments
sorted_total = top_total.sort_values('count_of_appointments')
axes[0].barh(sorted_total['icb_ons_name'], sorted_total['count_of_a
axes[0].set_title("Top 5 NHS ICBs by Total Appointments\n(Aug 2021 -
```

```
axes[0].set_xlabel("Number of Appointments")
for i, v in enumerate(sorted_total['count_of_appointments']):
    axes[0].text(v + 20000, i, f"{v:,.0f}", va='center', fontsize=9

# Per 1000 population
sorted_1000 = top_1000.sort_values('appointments_per_1000')
axes[1].barh(sorted_1000['icb_ons_name'], sorted_1000['appointments_
axes[1].set_title("Top 5 NHS ICBs by Appointments per 1,000\n(Aug 2(
axes[1].set_xlabel("Appointments per 1,000 People")
for i, v in enumerate(sorted_1000['appointments_per_1000']):
    axes[1].text(v + 50, i, f"{v:,.1f}", va='center', fontsize=9)

plt.tight_layout()
plt.show()
```



Question 3: How many service settings, context types, national categories, and appointment statuses are there?

In [42]:
```
# Determine the number of unique service settings
num_service_settings = nc['service_setting'].nunique()
print(f"Number of unique service settings: {num_service_settings}")
```

```
Number of unique service settings: 5
```

In [43]:
```
# List all unique service settings
unique_service_settings = nc['service_setting'].unique()
print("Unique service settings:\n")
print(unique_service_settings)
```

```
Unique service settings:

['Primary Care Network' 'Other' 'General Practice' 'Unmapped'
 'Extended Access Provision']
```

In [44]:
```
# Determine the number of unique context types
num_context_types = nc['context_type'].nunique()
print(f"Number of unique context types: {num_context_types}")

# List all unique context types
unique_context_types = nc['context_type'].unique()
print("Unique context types:\n")
```

```
print(unique_context_types)
```

```
Number of unique context types: 3
Unique context types:

['Care Related Encounter' 'Unmapped' 'Inconsistent Mapping']
```

In [45]:
```
# Determine the number of unique national categories
num_national_categories = nc['national_category'].nunique()
print(f"Number of unique national categories: {num_national_categor

# List all unique national categories
unique_national_categories = nc['national_category'].unique()
print("Unique national categories:\n")
print(unique_national_categories)
```

```
Number of unique national categories: 18
Unique national categories:

['Patient contact during Care Home Round' 'Planned Clinics' 'Home Vi
sit'
 'General Consultation Acute' 'Structured Medication Review'
 'Care Home Visit' 'Unmapped' 'Clinical Triage'
 'Planned Clinical Procedure' 'Inconsistent Mapping'
 'Care Home Needs Assessment & Personalised Care and Support Plannin
g'
 'General Consultation Routine'
 'Service provided by organisation external to the practice'
 'Unplanned Clinical Activity' 'Social Prescribing Service'
 'Non-contractual chargeable work'
 'Group Consultation and Group Education' 'Walk-in']
```

In [46]:
```
# Determine the number of unique appointment statuses
num_appointment_statuses = ar['appointment_status'].nunique()
print(f"Number of unique appointment statuses: {num_appointment_sta

# List all unique appointment statuses
unique_appointment_statuses = ar['appointment_status'].unique()
print("Unique appointment statuses:\n")
print(unique_appointment_statuses)
```

```
Number of unique appointment statuses: 3
Unique appointment statuses:

['Attended' 'DNA' 'Unknown']
```

In [47]:
```
# Load the national categories dataset
national_categories = pd.read_excel("national_categories.xlsx")

# Count unique service settings
service_settings_count = national_categories['service_setting'].nun

# Count unique context types
context_types_count = national_categories['context_type'].nunique()

# Count unique national categories
national_categories_count = national_categories['national_category'
```

```
# Count unique appointment statuses
appointment_statuses_count = appointments['appointment_status'].nun

# Print the results
print(f"Number of unique service settings: {service_settings_count}"
print(f"Number of unique context types: {context_types_count}")
print(f"Number of unique national categories: {national_categories_
print(f"Number of unique appointment statuses: {appointment_statuse
```

```
Number of unique service settings: 5
Number of unique context types: 3
Number of unique national categories: 18
Number of unique appointment statuses: 3
```

Assignment activity 3

Continue to explore the data and search for answers to more specific questions posed by the NHS.

Question 1: Between what dates were appointments scheduled?

In [51]:
```
# Ensure the appointment_date column is in datetime format
nc['appointment_date'] = pd.to_datetime(nc['appointment_date'], err

# Drop any rows with missing dates
nc = nc.dropna(subset=['appointment_date'])

# Define the earliest and latest appointment dates
min_date_nc = nc['appointment_date'].min()
max_date_nc = nc['appointment_date'].max()

# Output summary
print(f"Appointments were scheduled between {min_date_nc.date()} an
```

```
Appointments were scheduled between 2021-08-01 and 2022-06-30.
```

In [52]:
```
# Change the date format to YYYY-MM-DD (if not already in datetime)
ad['appointment_date'] = pd.to_datetime(ad['appointment_date'])

# Optional: if you want to store it as a string formatted as YYYY-M
ad['appointment_date_str'] = ad['appointment_date'].dt.strftime('%Y

# View the DataFrame (first 5 rows)
print(ad.head())
```

```
  sub_icb_location_code sub_icb_location_ons_code  \
0                   00L                 E38000130
1                   00L                 E38000130
2                   00L                 E38000130
3                   00L                 E38000130
4                   00L                 E38000130

                       sub_icb_location_name icb_ons_code region_on
s_code  \
0  NHS North East and North Cumbria ICB - 00L    E54000050       E40
000012
1  NHS North East and North Cumbria ICB - 00L    E54000050       E40
000012
2  NHS North East and North Cumbria ICB - 00L    E54000050       E40
000012
3  NHS North East and North Cumbria ICB - 00L    E54000050       E40
000012
4  NHS North East and North Cumbria ICB - 00L    E54000050       E40
000012

  appointment_date         actual_duration  count_of_appointments  \
0       2021-12-01           31-60 Minutes                    364
1       2021-12-01           21-30 Minutes                    619
2       2021-12-01            6-10 Minutes                   1698
3       2021-12-01  Unknown / Data Quality                   1277
4       2021-12-01           16-20 Minutes                    730

   duration_minutes appointment_date_str
0              45.0           2021-12-01
1              25.0           2021-12-01
2               8.0           2021-12-01
3               NaN           2021-12-01
4              18.0           2021-12-01
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/147
5378697.py:2: UserWarning: Could not infer format, so each element w
ill be parsed individually, falling back to `dateutil`. To ensure pa
rsing is consistent and as-expected, please specify a format.
  ad['appointment_date'] = pd.to_datetime(ad['appointment_date'])
```

Question 2:** Which service setting was the most popular for NHS North West London from 1 January to 1 June 2022?

In [54]:

```
#Which service setting reported the most appointments in North west
# Filter for NHS North West London ICB using icb_ons_code (E5400002
nc_subset = nc[
    (nc['icb_ons_code'] == 'E54000027') &
    (nc['appointment_date'] >= '2022-01-01') &
    (nc['appointment_date'] <= '2022-06-01')
]

# Check which service setting had the most appointments
service_counts = nc_subset.groupby('service_setting')['count_of_app
print("Service setting counts in North West London (Jan-Jun 2022):"
```

```
print(service_counts)
```

```
Service setting counts in North West London (Jan–Jun 2022):
service_setting
General Practice             4804239
Unmapped                      391106
Other                         152897
Primary Care Network          109840
Extended Access Provision      98159
Name: count_of_appointments, dtype: int64
```

Question 3: **Which month had the highest number of appointments?

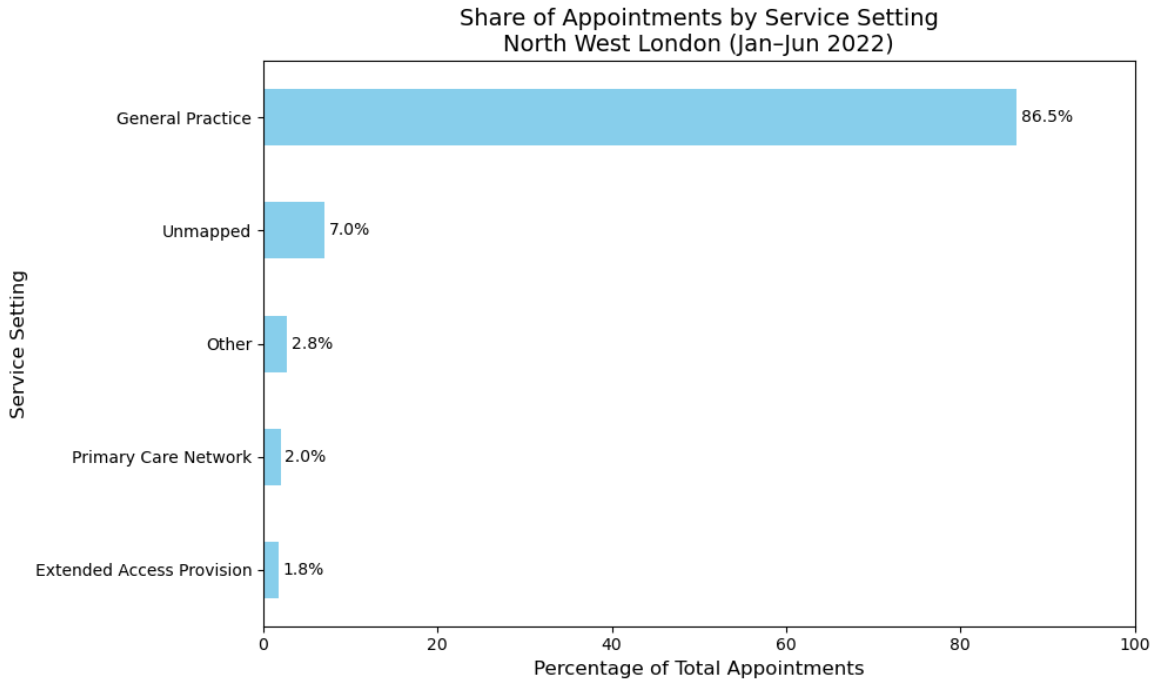
In [56]:
```
# Load datasets (if not already loaded)
ar = pd.read_csv("appointments_regional.csv")
nc = pd.read_excel("national_categories.xlsx")
ad = pd.read_csv("actual_duration.csv")

# Ensure datetime formats
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'], e
nc['appointment_date'] = pd.to_datetime(nc['appointment_date'], err
ad['appointment_date'] = pd.to_datetime(ad['appointment_date'], err

# Group totals per month
ar_monthly = ar.groupby(ar['appointment_month'].dt.to_period('M'))[
nc_monthly = nc.groupby(nc['appointment_date'].dt.to_period('M'))['
ad_monthly = ad.groupby(ad['appointment_date'].dt.to_period('M'))['

# Display results
print("Appointments Regional Data (Monthly Totals):")
print(ar_monthly)

print("\n National Categories Data (Monthly Totals):")
print(nc_monthly)

print("\n Actual Duration Data (Monthly Totals):")
print(ad_monthly)
```

```
Appointments Regional Data (Monthly Totals):
appointment_month
2020-01    27199296
2020-02    24104621
2020-03    24053468
2020-04    16007881
2020-05    16417212
2020-06    20690805
2020-07    22491437
2020-08    20150520
2020-09    26714255
2020-10    28301932
2020-11    25061602
2020-12    23535936
2021-01    22492069
2021-02    22399569
```

```
2021-03    27225424
2021-04    23879932
2021-05    23508395
2021-06    26784182
2021-07    25739219
2021-08    23852171
2021-09    28522501
2021-10    30303834
2021-11    30405070
2021-12    25140776
2022-01    25635474
2022-02    25355260
2022-03    29595038
2022-04    23913060
2022-05    27495508
2022-06    25828078
Freq: M, Name: count_of_appointments, dtype: int64

 National Categories Data (Monthly Totals):
appointment_date
2021-08    23852171
2021-09    28522501
2021-10    30303834
2021-11    30405070
2021-12    25140776
2022-01    25635474
2022-02    25355260
2022-03    29595038
2022-04    23913060
2022-05    27495508
2022-06    25828078
Freq: M, Name: count_of_appointments, dtype: int64

 Actual Duration Data (Monthly Totals):
appointment_date
2021-12    22853483
2022-01    23597196
2022-02    23351939
2022-03    27170002
2022-04    21948814
2022-05    25343941
2022-06    23715317
Freq: M, Name: count_of_appointments, dtype: int64
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/118
7881511.py:9: UserWarning: Could not infer format, so each element w
ill be parsed individually, falling back to `dateutil`. To ensure pa
rsing is consistent and as-expected, please specify a format.
  ad['appointment_date'] = pd.to_datetime(ad['appointment_date'], er
rors='coerce')
```

In [57]:
```
# Calculate percentage share of each service setting
service_counts_percent = (service_counts / service_counts.sum()) *

# Plot the percentage-based bar chart
plt.figure(figsize=(10, 6))
service_counts_percent.sort_values().plot(kind='barh', color='skybl
```

```
# Format chart
plt.title('Share of Appointments by Service Setting\nNorth West Lon
plt.xlabel('Percentage of Total Appointments', fontsize=12)
plt.ylabel('Service Setting', fontsize=12)
plt.xlim(0, 100)

# Add percentage labels
for index, value in enumerate(service_counts_percent.sort_values())
    plt.text(value + 0.5, index, f"{value:.1f}%", va='center')

plt.tight_layout()
plt.show()
```



Monthly Appointment Trends: NHS Data (2020–2022)

This chart below compares the total number of appointments recorded per month in two key NHS datasets:

- **Appointments Regional**: Complete dataset spanning from January 2020 to June 2022.
- **National Categories**: Available from August 2021 onward.

Key Observations:

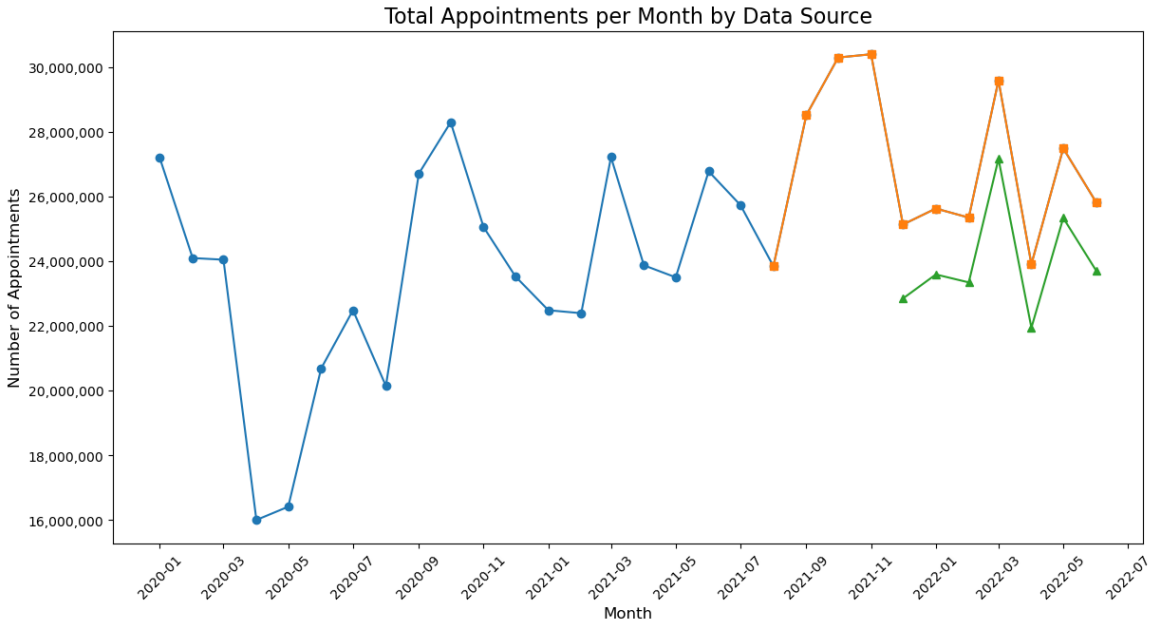
- From August 2021 onward, both datasets report **identical monthly totals**, confirming data consistency.
- The highest monthly volume was observed in **November 2021**, reaching over **30 million** appointments.
- A sharp decline occurred in **April 2020**, likely due to pandemic-related disruptions.

- The grey dashed line marks the **average monthly volume** in the regional dataset (approx. 25 million).

This visualisation helps stakeholders understand the evolution of NHS appointment activity, verify data alignment between sources, and identify seasonal or exceptional peaks.

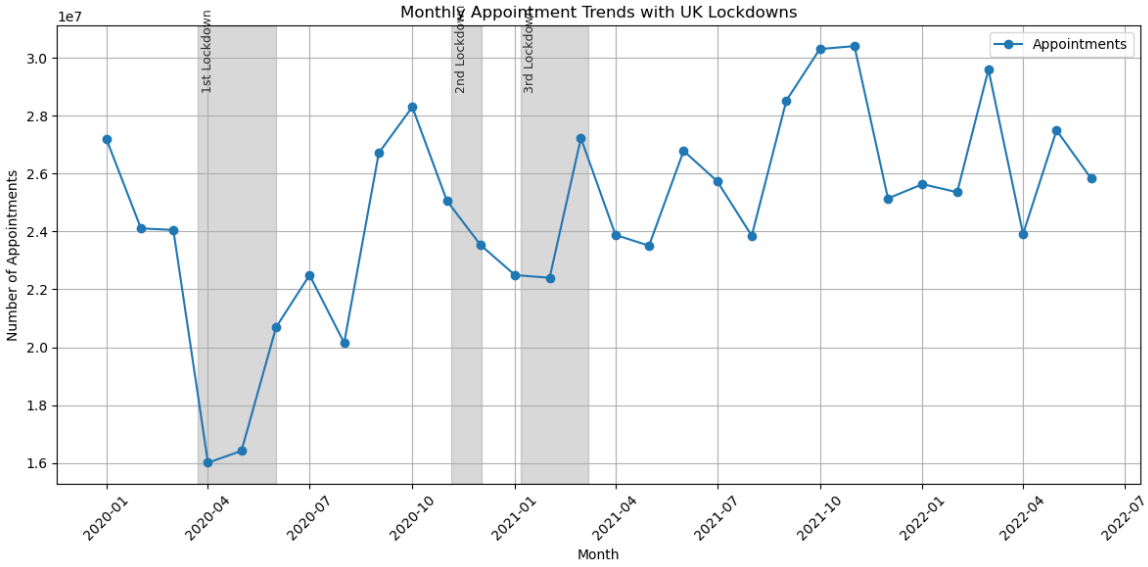
In [59]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import matplotlib.dates as mdates
import pandas as pd

# Convert PeriodIndex to Timestamp
ar_monthly.index = ar_monthly.index.to_timestamp()
nc_monthly.index = nc_monthly.index.to_timestamp()
ad_monthly.index = ad_monthly.index.to_timestamp()

# Plot
plt.figure(figsize=(14, 7))
plt.plot(ar_monthly.index, ar_monthly.values, label='Appointments R
plt.plot(nc_monthly.index, nc_monthly.values, label='National Categ
plt.plot(ad_monthly.index, ad_monthly.values, label='Actual Duratio

# Title and Axis Labels
plt.title("Total Appointments per Month by Data Source", fontsize=1
plt.xlabel("Month", fontsize=12)
plt.ylabel("Number of Appointments", fontsize=12)

# Format Y-axis with commas
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _

# Format X-axis
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=2))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
plt.xticks(rotation=45)

# Grid and legend
plt.grid
```

Out[59]:
```
<function matplotlib.pyplot.grid(visible: 'bool | None' = None, wh
ich: "Literal['major', 'minor', 'both']" = 'major', axis: "Litera
l['both', 'x', 'y']" = 'both', **kwargs) -> 'None'>
```

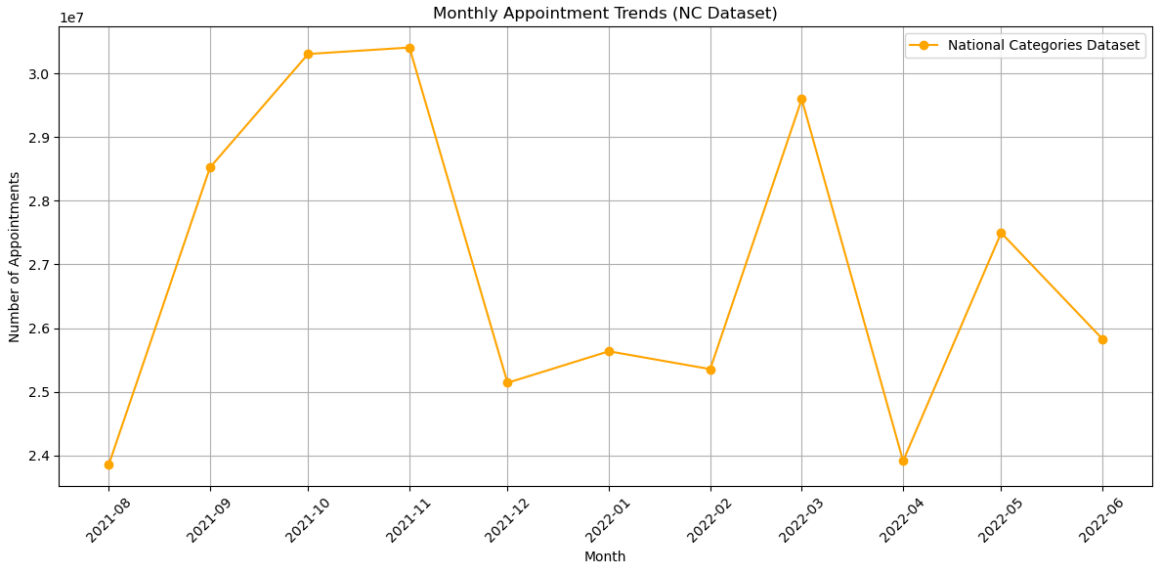


In [60]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Ensure the 'appointment_month' column is datetime
appointments['appointment_month'] = pd.to_datetime(appointments['ap

# Define lockdown periods
lockdowns = [
    ('2020-03-23', '2020-06-01', '1st Lockdown'),
    ('2020-11-05', '2020-12-02', '2nd Lockdown'),
    ('2021-01-06', '2021-03-08', '3rd Lockdown')
]

# Group appointments by month
appointments_monthly = appointments.groupby(
    appointments['appointment_month'].dt.to_period('M')
)['count_of_appointments'].sum()

# Convert PeriodIndex to datetime
appointments_monthly.index = appointments_monthly.index.to_timestam

# Plot the line chart
plt.figure(figsize=(12, 6))
plt.plot(appointments_monthly.index, appointments_monthly, marker='

# Add shaded lockdown periods
for start, end, label in lockdowns:
    plt.axvspan(pd.to_datetime(start), pd.to_datetime(end), color='
    plt.text(pd.to_datetime(start) + pd.Timedelta(days=3),
             appointments_monthly.max() * 0.95,
             label, rotation=90, color='black', fontsize=9, alpha=0

plt.title('Monthly Appointment Trends with UK Lockdowns')
plt.ylabel('Number of Appointments')
plt.xlabel('Month')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
```

```
plt.tight_layout()
plt.show()
```



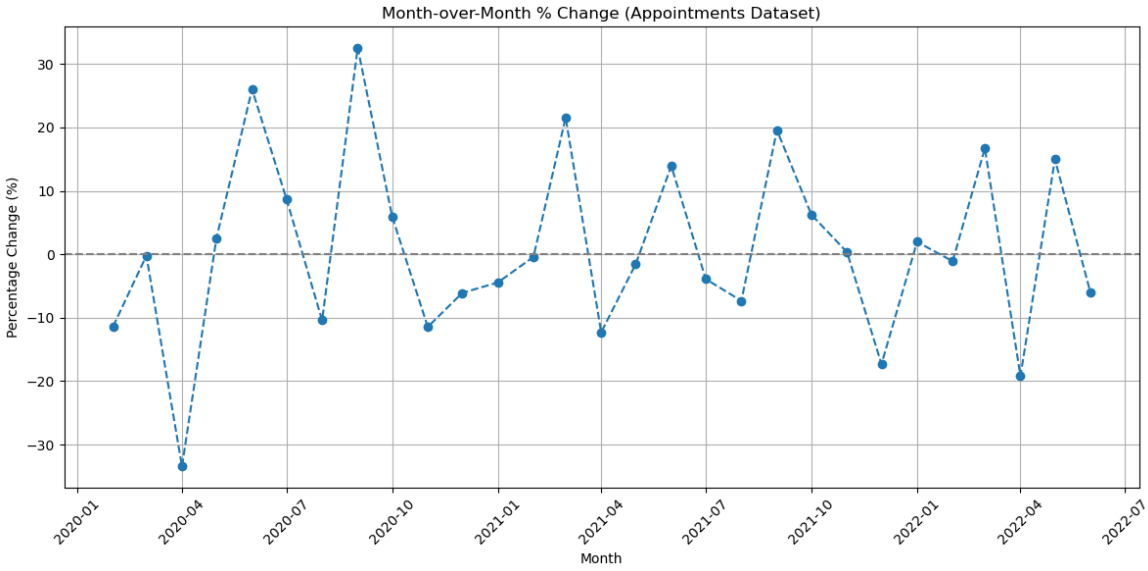
In [61]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Group by month and convert period index to timestamp
nc_monthly = nc.groupby(
    nc['appointment_date'].dt.to_period('M')
)['count_of_appointments'].sum()
nc_monthly.index = nc_monthly.index.to_timestamp()

# Plot without lockdown shading
plt.figure(figsize=(12, 6))
plt.plot(nc_monthly.index, nc_monthly, marker='o', color='orange',

plt.title('Monthly Appointment Trends (NC Dataset)')
plt.ylabel('Number of Appointments')
plt.xlabel('Month')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

# note (as markdown or in report)
print("*Note: This dataset begins after the final UK lockdown perio
```



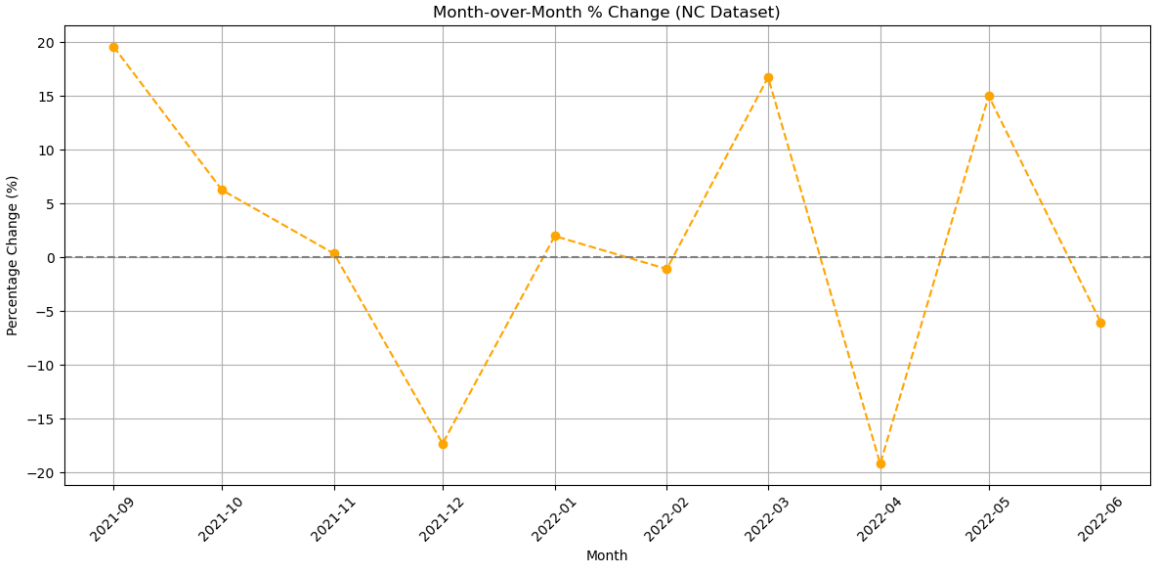
Note: This dataset begins after the final UK lockdown period.

In [62]:
```
# Appointments dataset
appointments_pct_change = appointments_monthly.pct_change() * 100
print("Appointments dataset - Month-over-month percentage change:")
print(appointments_pct_change)

# Plot
plt.figure(figsize=(12,6))
plt.plot(appointments_pct_change.index, appointments_pct_change, ma
plt.title('Month-over-Month % Change (Appointments Dataset)')
plt.ylabel('Percentage Change (%)')
plt.xlabel('Month')
plt.xticks(rotation=45)
plt.grid(True)
plt.axhline(0, color='gray', linestyle='--')
plt.tight_layout()
plt.show()

# NC dataset
nc_pct_change = nc_monthly.pct_change() * 100
print("National Categories dataset - Month-over-month percentage ch
print(nc_pct_change)

# Plot
plt.figure(figsize=(12,6))
plt.plot(nc_pct_change.index, nc_pct_change, marker='o', linestyle=
plt.title('Month-over-Month % Change (NC Dataset)')
plt.ylabel('Percentage Change (%)')
plt.xlabel('Month')
plt.xticks(rotation=45)
plt.grid(True)
plt.axhline(0, color='gray', linestyle='--')
plt.tight_layout()
plt.show()
```

```
Appointments dataset – Month-over-month percentage change:
appointment_month
2020-01-01          NaN
2020-02-01   -11.377776
2020-03-01    -0.212212
2020-04-01   -33.448761
2020-05-01     2.557059
2020-06-01    26.031174
2020-07-01     8.702571
2020-08-01   -10.408037
2020-09-01    32.573527
2020-10-01     5.943183
2020-11-01   -11.449148
2020-12-01    -6.087664
2021-01-01    -4.435205
2021-02-01    -0.411256
2021-03-01    21.544410
2021-04-01   -12.288117
2021-05-01    -1.555855
2021-06-01    13.934541
2021-07-01    -3.901418
2021-08-01    -7.331411
2021-09-01    19.580314
2021-10-01     6.245360
2021-11-01     0.334070
2021-12-01   -17.313869
2022-01-01     1.967712
2022-02-01    -1.093071
2022-03-01    16.721493
2022-04-01   -19.199090
2022-05-01    14.981136
2022-06-01    -6.064372
Freq: MS, Name: count_of_appointments, dtype: float64
```



```
National Categories dataset - Month-over-month percentage change:
appointment_date
2021-08-01          NaN
2021-09-01    19.580314
2021-10-01     6.245360
2021-11-01     0.334070
2021-12-01   -17.313869
2022-01-01     1.967712
2022-02-01    -1.093071
2022-03-01    16.721493
2022-04-01   -19.199090
2022-05-01    14.981136
2022-06-01    -6.064372
Freq: MS, Name: count_of_appointments, dtype: float64
```



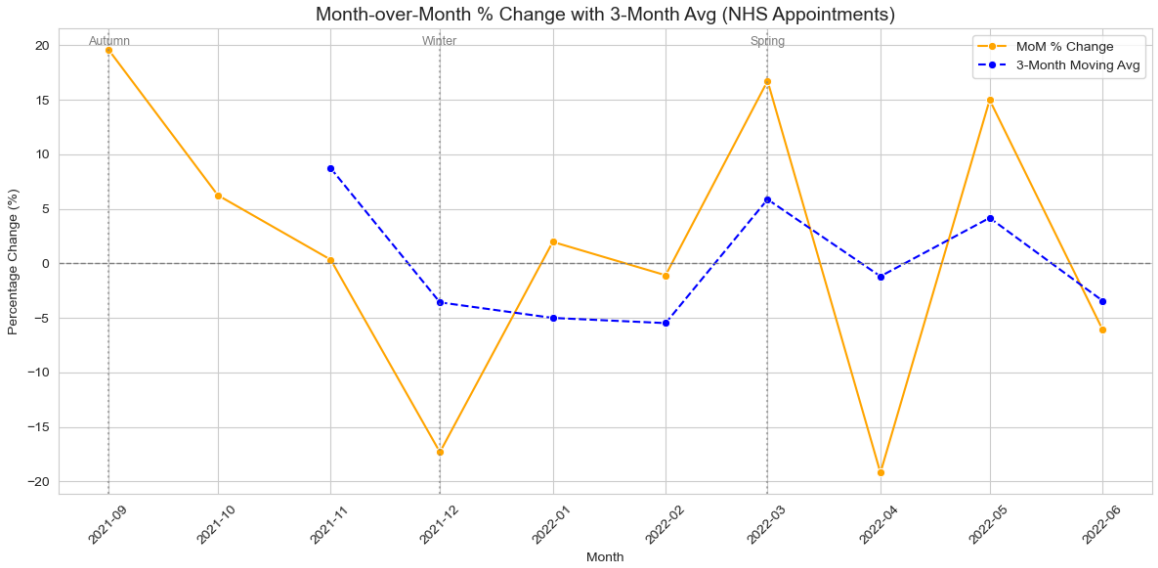
In [63]:
```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv("appointments_regional.csv")

# Convert appointment_month to datetime
df['appointment_month'] = pd.to_datetime(df['appointment_month'])

# Filter from August 2021 to June 2022
df = df[(df['appointment_month'] >= '2021-08-01') & (df['appointmen

# Group by month and sum total appointments
monthly_df = df.groupby('appointment_month')['count_of_appointments

# Calculate month-over-month % change
monthly_df['pct_change'] = monthly_df['count_of_appointments'].pct_

# Calculate 3-month moving average of % change
monthly_df['3mo_avg'] = monthly_df['pct_change'].rolling(window=3).

# Display table
monthly_df[['appointment_month', 'count_of_appointments', 'pct_chan
```

Out[63]:

	appointment_month	count_of_appointments	pct_change	3mo_avg
0	2021-08-01	23852171	NaN	NaN
1	2021-09-01	28522501	19.580314	NaN
2	2021-10-01	30303834	6.245360	NaN
3	2021-11-01	30405070	0.334070	8.719915
4	2021-12-01	25140776	-17.313869	-3.578146
5	2022-01-01	25635474	1.967712	-5.004029
6	2022-02-01	25355260	-1.093071	-5.479743
7	2022-03-01	29595038	16.721493	5.865378
8	2022-04-01	23913060	-19.199090	-1.190223
9	2022-05-01	27495508	14.981136	4.167846
10	2022-06-01	25828078	-6.064372	-3.427442

In [64]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot MoM % Change and 3-Month Avg
plt.figure(figsize=(12, 6))
sns.set_style("whitegrid")

# MoM % Change Line
sns.lineplot(data=monthly_df, x='appointment_month', y='pct_change'
             marker='o', color='orange', label='MoM % Change')

# 3-Month Average Line
sns.lineplot(data=monthly_df, x='appointment_month', y='3mo_avg',
             marker='o', linestyle='--', color='blue', label='3-Mon

# Horizontal reference at 0%
plt.axhline(0, linestyle='--', color='gray', linewidth=1)

# Add seasonal markers
for season, month in [('Autumn', '2021-09'), ('Winter', '2021-12'),
    plt.axvline(pd.to_datetime(month), color='gray', linestyle=':',
    plt.text(pd.to_datetime(month), 20, season, fontsize=9, color='

# Titles and labels
plt.title("Month-over-Month % Change with 3-Month Avg (NHS Appointm
plt.xlabel("Month")
plt.ylabel("Percentage Change (%)")
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



Insights for Stakeholders

Seasonal Demand Peaks:

→ November 2021 saw the highest appointment load across both the appointments and national categories datasets. → This suggests a potential surge in healthcare demand late in the year (likely linked to seasonal illnesses like flu, catch-ups after COVID waves, or winter pressures).

Shift in Duration Data:

→ The actual duration data peaks in March 2022, possibly because duration tracking only started later (as the metadata notes it was first included from December 2021). → So, the March peak reflects the new recording practice rather than a real surge — this is important context for interpretation.

Question 4: What was the total number of records per month?

In [67]:
```
import pandas as pd

# Load datasets
appointments = pd.read_csv('appointments_regional.csv')
national_categories = pd.read_excel('national_categories.xlsx')
actual_duration = pd.read_csv('actual_duration.csv')

# Ensure date columns are parsed correctly
appointments['appointment_month'] = pd.to_datetime(appointments['ap
national_categories['appointment_month'] = pd.to_datetime(national_
actual_duration['appointment_date'] = pd.to_datetime(actual_duratio

# Count records (rows) per month for each dataset
appointments_records_per_month = appointments.groupby(appointments[
nc_records_per_month = national_categories.groupby(national_categor
```

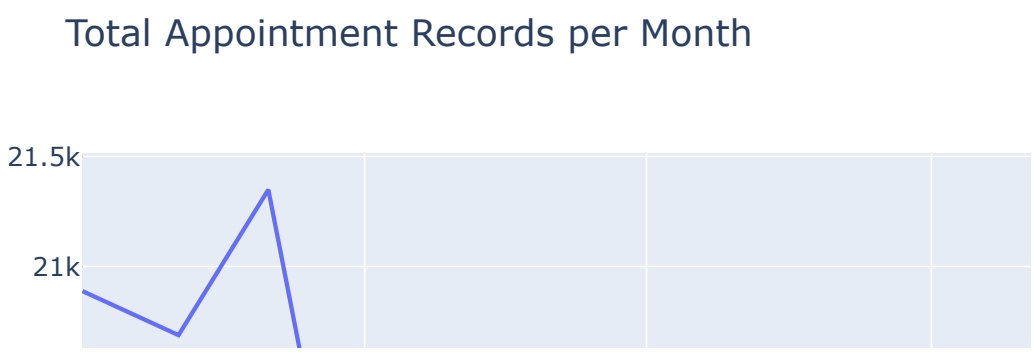
```
ad_records_per_month = actual_duration.groupby(actual_duration['app

# Print results
print("Appointments Regional Data: Total records per month")
print(appointments_records_per_month)
print("\n National Categories Data: Total records per month")
print(nc_records_per_month)
print("\nActual Duration Data: Total records per month")
print(ad_records_per_month)
```

```
Appointments Regional Data: Total records per month
appointment_month
2020-01    20889
2020-02    20689
2020-03    21350
2020-04    19124
2020-05    18338
2020-06    18844
2020-07    19502
2020-08    19247
2020-09    20043
2020-10    20122
2020-11    19675
2020-12    19394
2021-01    19319
2021-02    18949
2021-03    19369
2021-04    19452
2021-05    19384
2021-06    19814
2021-07    19899
2021-08    19786
2021-09    20441
2021-10    20562
2021-11    20766
2021-12    20393
2022-01    20225
2022-02    20133
2022-03    20532
2022-04    20073
2022-05    20276
2022-06    20231
Freq: M, dtype: int64

 National Categories Data: Total records per month
appointment_month
2021-08    69999
2021-09    74922
2021-10    74078
2021-11    77652
2021-12    72651
2022-01    71896
2022-02    71769
2022-03    82822
2022-04    70012
2022-05    77425
```

```
2022-06    74168
Freq: M, dtype: int64

Actual Duration Data: Total records per month
appointment_date
2021-12    19507
2022-01    19643
2022-02    18974
2022-03    21236
2022-04    19078
2022-05    20128
2022-06    19227
Freq: M, dtype: int64
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/197
7572529.py:11: UserWarning: Could not infer format, so each element
will be parsed individually, falling back to `dateutil`. To ensure p
arsing is consistent and as-expected, please specify a format.
  actual_duration['appointment_date'] = pd.to_datetime(actual_durati
on['appointment_date'], errors='coerce')
```

In [68]:
```
monthly_records = appointments.groupby(appointments['appointment_mo
monthly_records['appointment_month'] = monthly_records['appointment_

import plotly.express as px
fig = px.line(monthly_records, x='appointment_month', y='total_reco
fig.show()
```



What was the total number of records per month?

We calculated the total number of records (rows) per month across the three datasets:

Appointments Regional Data (2020–2022):

- Monthly records: ~18,000–21,000
- Coverage: From January 2020 through June 2022
- Insight: Stable monthly record counts, suggesting consistent data reporting across time.

National Categories Data (2021–2022):

- Monthly records: ~70,000–83,000
- Coverage: From August 2021 through June 2022
- Insight: Higher monthly record counts, starting later than the regional

data; consistent reporting once available.

Actual Duration Data (2021–2022):

- Monthly records: ~19,000–21,000
- Coverage: From December 2021 through June 2022
- Insight: Similar record volume to the regional data but with a shorter available period (aligned with when duration tracking was first included).

Key Takeaways:

- All three datasets show steady and consistent monthly record volumes.
- Later-starting datasets (like Actual Duration) reflect when new data types were first introduced.
- There are no obvious missing or drastically low months, which suggests the datasets are reliable for further monthly or seasonal analysis.

Assignment activity 4

Create visualisations and identify possible monthly and seasonal trends in the data.

In [73]:
```
# Import the necessary libraries.
import seaborn as sns
import matplotlib.pyplot as plt

# Set figure size.
sns.set(rc={'figure.figsize':(15, 12)})

# Set the plot style as white.
sns.set_style('white')
```

Objective 1 Summary: Monthly Trends

We analyzed the number of appointments per month across service settings, context types, and national categories.

Key observations:

- **Service settings**:
  - General Practice had consistently the highest number of

appointments.
    - Other settings like Extended Access Provision and Primary Care Network showed lower but steady counts.
- **Context types**:
    - The majority of appointments were classified under Care Related Encounter.
    - Unmapped and Inconsistent Mapping categories had much lower volumes, which may signal data quality issues.
- **National categories**:
    - General Consultation Acute and Routine dominated monthly appointment counts.
    - Categories like Clinical Triage and Home Visits remained smaller but stable over time.

Insights:

- There was a noticeable peak in appointments around March 2022.
- Understanding these monthly patterns can help NHS plan resources and staff scheduling more effectively.

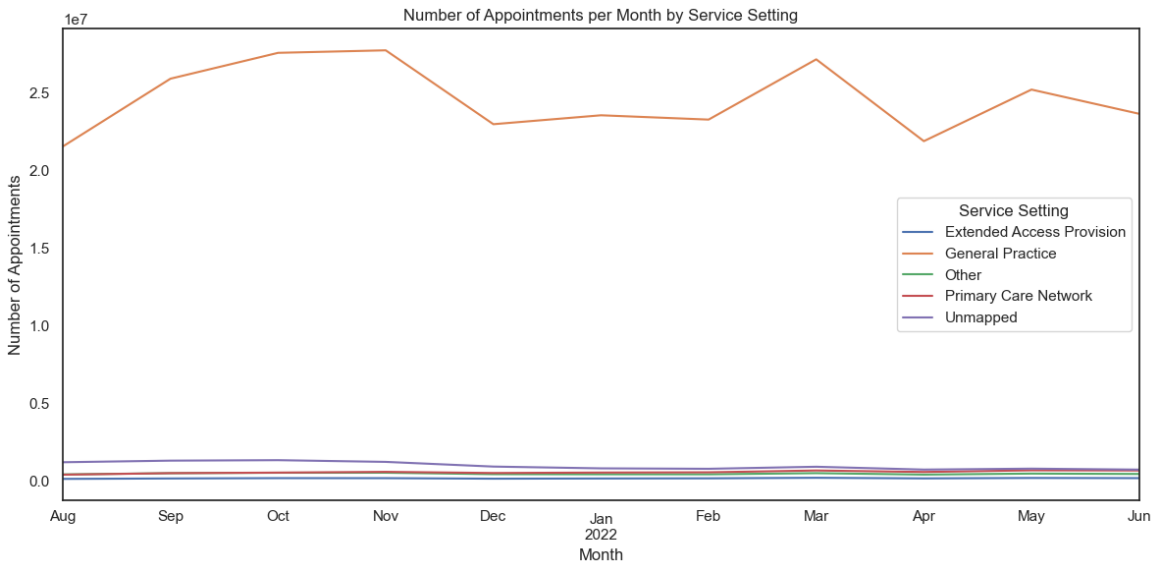
We visualized these trends using Seaborn lineplots with month on the x-axis and appointment counts on the y-axis, colored by category (`hue`).

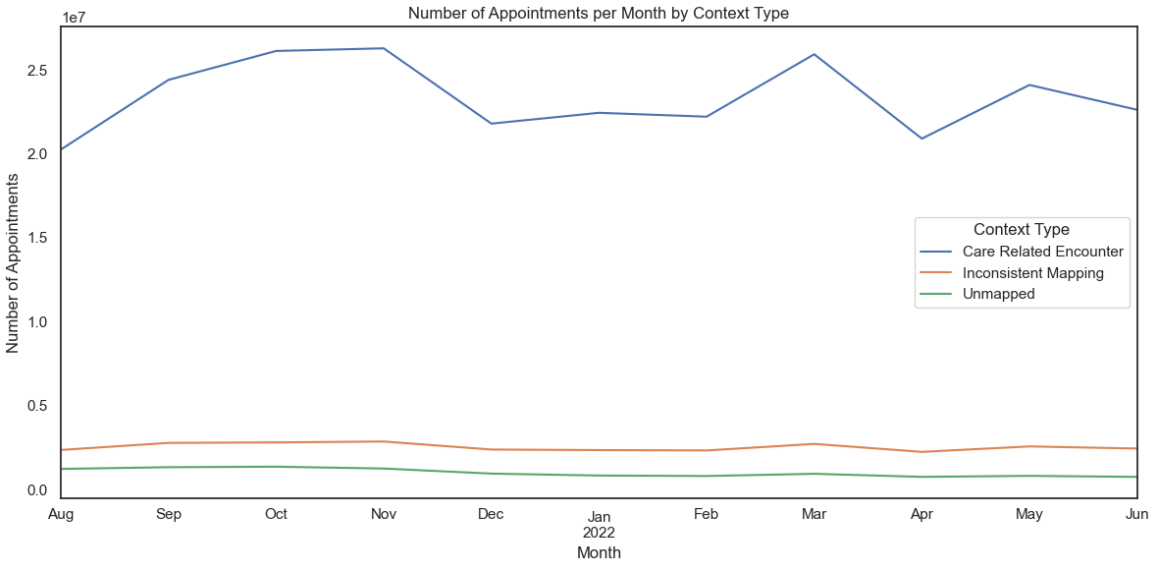
Objective 1

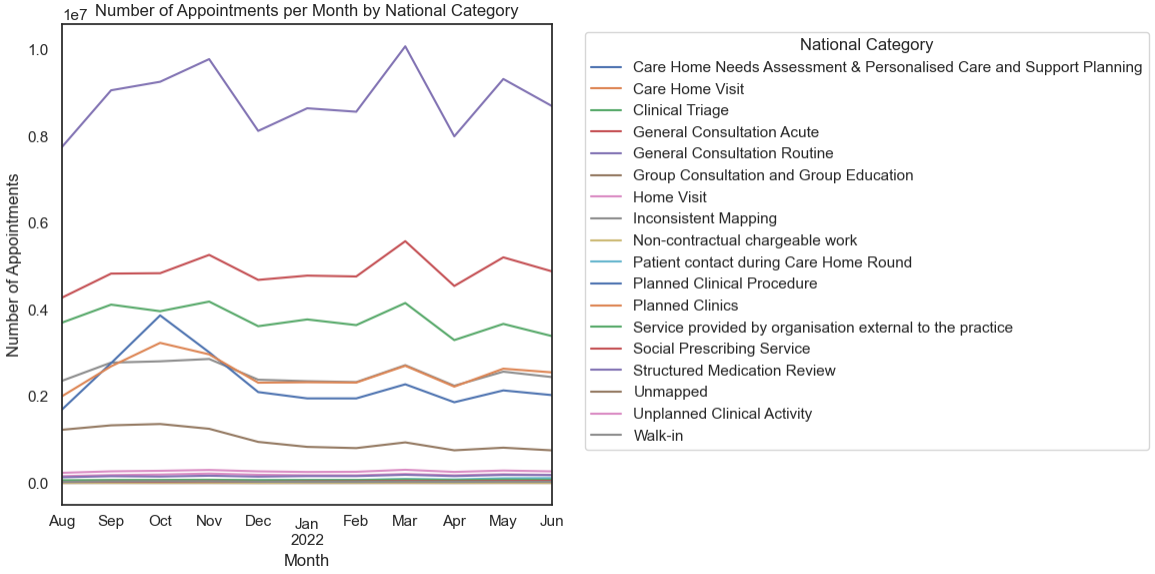
Create three visualisations indicating the number of appointments per month for service settings, context types, and national categories.

In [76]:
```
# Load data
national_categories = pd.read_excel('national_categories.xlsx')

# Ensure date column is datetime
national_categories['appointment_month'] = pd.to_datetime(national_

# Group by month + service setting
service_setting_summary = national_categories.groupby([
    national_categories['appointment_month'].dt.to_period('M'),
    'service_setting'
])['count_of_appointments'].sum().unstack()

# Group by month + context type
context_type_summary = national_categories.groupby([
    national_categories['appointment_month'].dt.to_period('M'),
    'context_type'
])['count_of_appointments'].sum().unstack()
```

```
# Group by month + national category
national_category_summary = national_categories.groupby([
    national_categories['appointment_month'].dt.to_period('M'),
    'national_category'
])['count_of_appointments'].sum().unstack()

# Plot 1: Service Setting
service_setting_summary.plot(kind='line', figsize=(12,6))
plt.title('Number of Appointments per Month by Service Setting')
plt.xlabel('Month')
plt.ylabel('Number of Appointments')
plt.legend(title='Service Setting')
plt.tight_layout()
plt.show()

# Plot 2: Context Type
context_type_summary.plot(kind='line', figsize=(12,6))
plt.title('Number of Appointments per Month by Context Type')
plt.xlabel('Month')
plt.ylabel('Number of Appointments')
plt.legend(title='Context Type')
plt.tight_layout()
plt.show()

# Plot 3: National Category
national_category_summary.plot(kind='line', figsize=(12,6))
plt.title('Number of Appointments per Month by National Category')
plt.xlabel('Month')
plt.ylabel('Number of Appointments')
plt.legend(title='National Category', bbox_to_anchor=(1.05, 1), loc
plt.tight_layout()
plt.show()
```







Objective 2

Create four visualisations indicating the number of appointments for service setting per season. The seasons are summer (June to August 2021), autumn (September to November 2021), winter (December to February 2022), and spring (March to May 2022).

Seasonal NHS Appointment Volumes (Aug 2021 – Jun 2022)

The chart below illustrates the total number of NHS appointments recorded during each defined seasonal period. These periods were constructed based on the availability of data and reflect the following groupings:

- **August 2021** (Standalone Summer Month)
- **Autumn 2021**: September, October, November
- **Winter 2021–2022**: December, January, February
- **Spring 2022**: March, April, May

- **June 2022** (Standalone Summer Month)

Key Findings:

- **Autumn 2021** saw the **highest appointment volume** across all seasons with **89,231,405** appointments.
- **Spring 2022** followed with **81,003,606** appointments.
- **Winter 2021–2022** recorded **76,131,510** appointments.
- **Standalone months**:
  - **August 2021**: 23,852,171 appointments
  - **June 2022**: 25,828,078 appointments

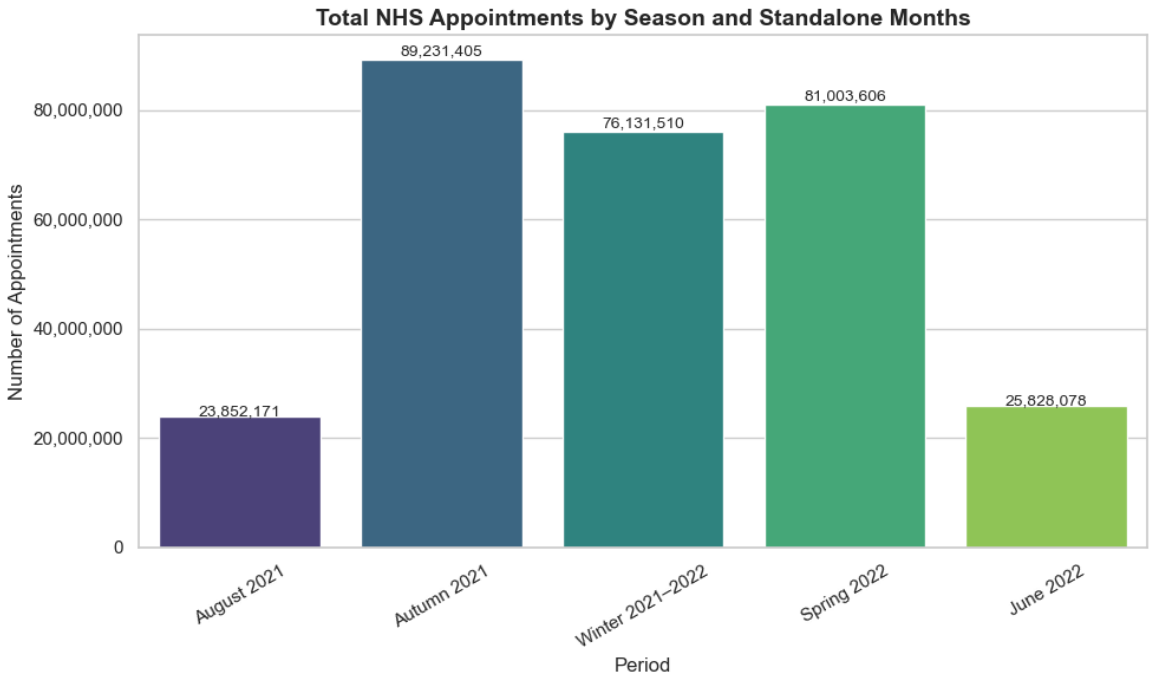
These figures reflect high service demand in the Autumn and Spring months, which may correspond with post-summer and pre-summer increases in healthcare utilisation.

> *Insight for stakeholders*: Seasonal fluctuations in appointments suggest potential planning opportunities for workforce and resource allocation, especially ahead of peak months such as October and March.

In [79]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure appointment_date is datetime
nc['appointment_date'] = pd.to_datetime(nc['appointment_date'])
nc['year_month'] = nc['appointment_date'].dt.to_period('M')

# Define custom periods
period_definitions = {
    'August 2021': ['2021-08'],
    'Autumn 2021': ['2021-09', '2021-10', '2021-11'],
    'Winter 2021–2022': ['2021-12', '2022-01', '2022-02'],
    'Spring 2022': ['2022-03', '2022-04', '2022-05'],
    'June 2022': ['2022-06']
}

# Aggregate total appointments per period
period_totals = []

for period, months in period_definitions.items():
    period_data = nc[nc['year_month'].astype(str).isin(months)]
    total_appointments = period_data['count_of_appointments'].sum()
    period_totals.append({'Period': period, 'Total_Appointments': t

# Create DataFrame
period_df = pd.DataFrame(period_totals)

# Sort periods in logical order
```

```
period_order = ['August 2021', 'Autumn 2021', 'Winter 2021–2022', '
period_df['Period'] = pd.Categorical(period_df['Period'], categorie
period_df = period_df.sort_values('Period')

# Plotting
plt.figure(figsize=(10, 6))
sns.set_style('whitegrid')

barplot = sns.barplot(
    data=period_df,
    x='Period',
    y='Total_Appointments',
    palette='viridis'
)

# Add title and labels
barplot.set_title('Total NHS Appointments by Season and Standalone I
barplot.set_ylabel('Number of Appointments', fontsize=12)
barplot.set_xlabel('Period', fontsize=12)

# Add value labels on bars
for i, row in period_df.iterrows():
    barplot.text(i, row['Total_Appointments'] + 0.01 * row['Total_Ap
                 f"{row['Total_Appointments']:,}", ha='center', fon

# Format y-axis with comma separator
barplot.set_yticklabels(['{:,}'.format(int(tick)) for tick in barpl

plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/410
1326328.py:38: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.


/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/410
1326328.py:56: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks,
i.e. after set_ticks() or using a FixedLocator.
```



Average Monthly NHS Appointments by Season (Normalised)

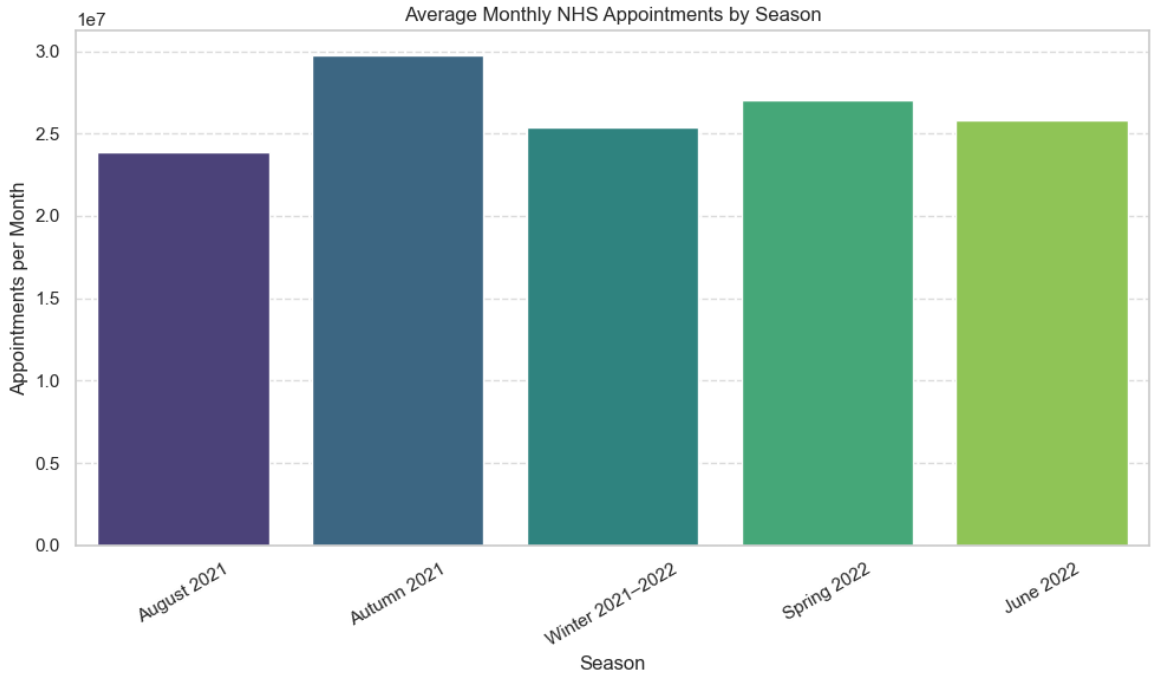
To allow for fair comparison across seasons with varying data coverage, the total number of appointments for each season was divided by the number of months included in that period:

August 2021 and June 2022 each represent only 1 month, Autumn, Winter, and Spring each represent 3 full months. This adjustment helps avoid misinterpreting seasonal trends due to unequal durations. For instance, while Autumn 2021 shows the highest overall volume of appointments, Spring 2022 has a comparable number per month, suggesting sustained demand after the winter period. The normalised values provide a more accurate understanding of service load and seasonal variation in demand.

In [81]:
```
import pandas as pd

# Manually recreate the season_df with your known values
season_df = pd.DataFrame({
    'Season': ['August 2021', 'Autumn 2021', 'Winter 2021–2022', 'Sp
    'Total_Appointments': [23852171, 89231405, 76131510, 81003606, 2
})

# Add a new column to divide by number of months
season_df['Months_Covered'] = [1, 3, 3, 3, 1]  # Example: Aug 2021
season_df['Appointments_per_Month'] = season_df['Total_Appointments

# Sort by date order
season_df = season_df.set_index('Season').loc[
    ['August 2021', 'Autumn 2021', 'Winter 2021–2022', 'Spring 2022
].reset_index()
```

```
# Plot normalized values
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.barplot(data=season_df, x='Season', y='Appointments_per_Month',

plt.title("Average Monthly NHS Appointments by Season")
plt.ylabel("Appointments per Month")
plt.xticks(rotation=30)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/917
605893.py:23: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.
```



In [82]:
```
print(nc.columns)
df = pd.read_excel("national_categories.xlsx")  # or pd.read_csv("a
print(df.columns)
```

```
Index(['appointment_date', 'icb_ons_code', 'sub_icb_location_name',
       'service_setting', 'context_type', 'national_category',
       'count_of_appointments', 'appointment_month', 'year_month'],
      dtype='object')
Index(['appointment_date', 'icb_ons_code', 'sub_icb_location_name',
       'service_setting', 'context_type', 'national_category',
       'count_of_appointments', 'appointment_month'],
      dtype='object')
```

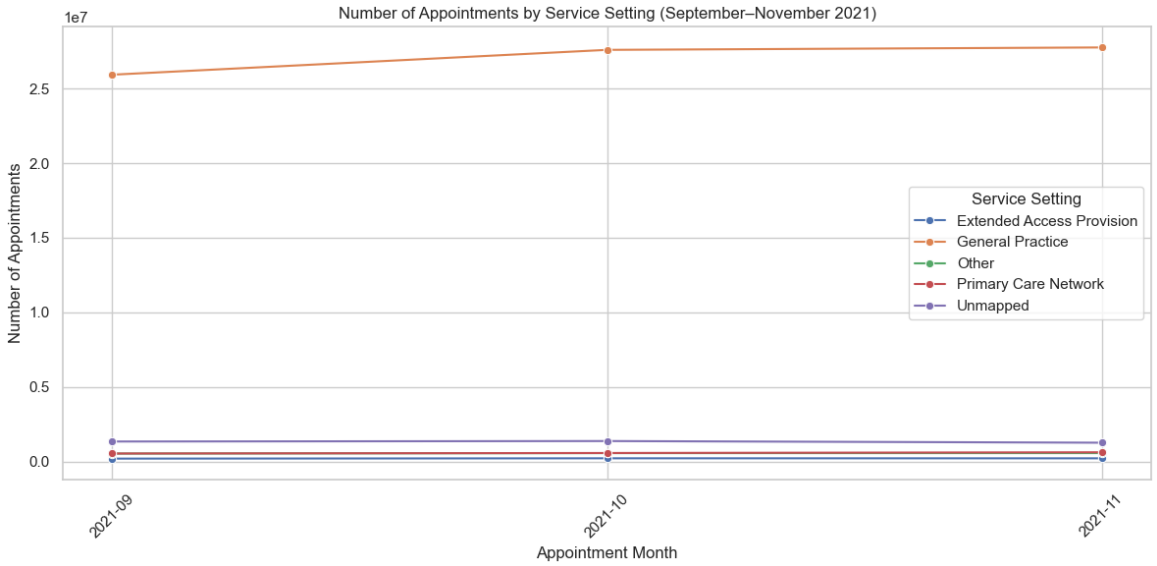
In [83]:
```
# Step 1: Create year_month from 'appointment_month'
df['appointment_month'] = pd.to_datetime(df['appointment_month'])
df['year_month'] = df['appointment_month'].dt.to_period('M').astype

# Step 2: Group by year_month to get raw monthly totals
monthly_totals = df.groupby('year_month')['count_of_appointments'].
monthly_totals.columns = ['YearMonth', 'Monthly_Total']

print("\nRaw Monthly Totals:")
print(monthly_totals)

# Step 3: Define seasonal periods
period_definitions = {
    'August 2021': ['2021-08'],
    'Autumn 2021': ['2021-09', '2021-10', '2021-11'],
    'Winter 2021-2022': ['2021-12', '2022-01', '2022-02'],
    'Spring 2022': ['2022-03', '2022-04', '2022-05'],
    'June 2022': ['2022-06']
}

# Step 4: Calculate totals per period
for period, months in period_definitions.items():
    period_sum = monthly_totals[monthly_totals['YearMonth'].isin(mo
    print(f"{period}: {period_sum:,} appointments")
```

```
Raw Monthly Totals:
   YearMonth  Monthly_Total
0    2021-08       23852171
1    2021-09       28522501
2    2021-10       30303834
3    2021-11       30405070
4    2021-12       25140776
5    2022-01       25635474
6    2022-02       25355260
7    2022-03       29595038
8    2022-04       23913060
9    2022-05       27495508
10   2022-06       25828078
August 2021: 23,852,171 appointments
Autumn 2021: 89,231,405 appointments
Winter 2021-2022: 76,131,510 appointments
Spring 2022: 81,003,606 appointments
June 2022: 25,828,078 appointments
```

Autumn (September to November 2021):

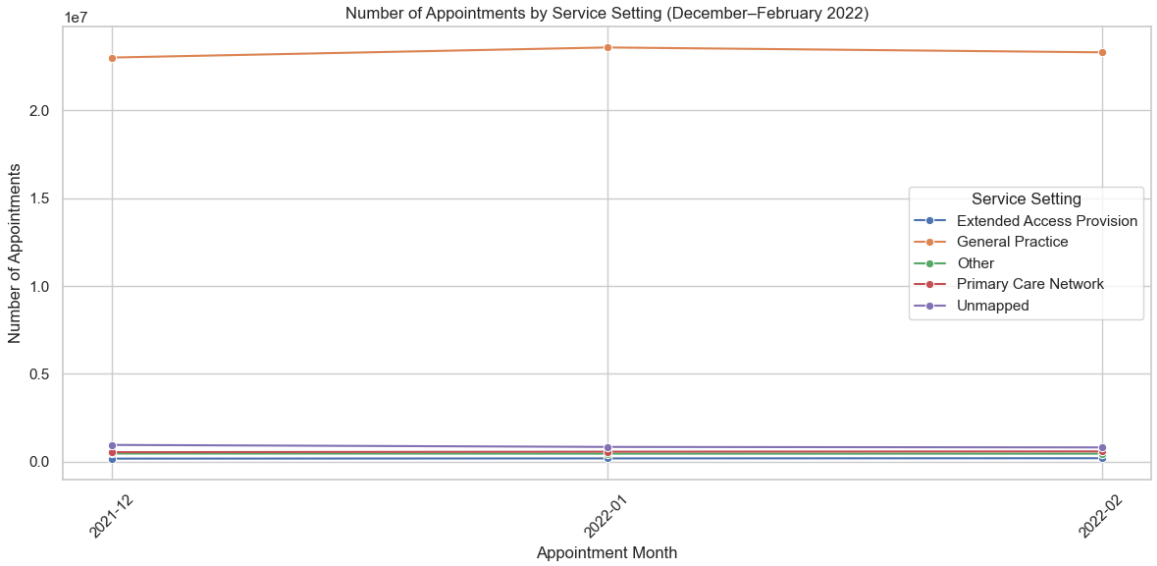
In [85]:
```
# Look at September to November 2021 in more detail to allow a clos
```

```
# Create a lineplot.
# Autumn: September–November 2021
autumn_nc = nc[nc['appointment_month'].isin(['2021-09', '2021-10',
autumn_summary = autumn_nc.groupby(['appointment_month', 'service_s

plt.figure(figsize=(12, 6))
sns.lineplot(data=autumn_summary, x='appointment_month', y='count_o

plt.title('Number of Appointments by Service Setting (September–Nov
plt.xlabel('Appointment Month')
plt.ylabel('Number of Appointments')
plt.xticks(rotation=45)
plt.legend(title='Service Setting')
plt.tight_layout()
plt.show()
```



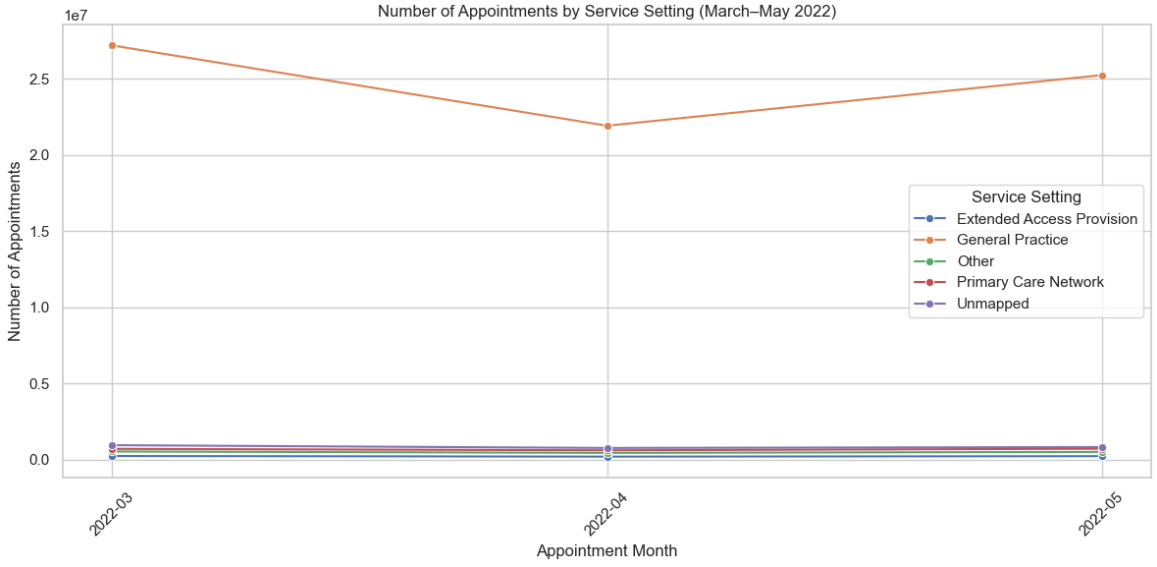
Winter (December to February 2022):

In [87]:
```
# Look at December to February 2022 in more detail to allow a close
# Create a lineplot.
# Winter: December–February 2022
winter_nc = nc[nc['appointment_month'].isin(['2021-12', '2022-01',
winter_summary = winter_nc.groupby(['appointment_month', 'service_s

plt.figure(figsize=(12, 6))
sns.lineplot(data=winter_summary, x='appointment_month', y='count_o

plt.title('Number of Appointments by Service Setting (December–Febr
plt.xlabel('Appointment Month')
plt.ylabel('Number of Appointments')
plt.xticks(rotation=45)
plt.legend(title='Service Setting')
plt.tight_layout()
plt.show()
```



Spring (March to May 2022):

In [89]:
```
# Look at March to May 2022 in more detail to allow a closer look.
# Create a lineplot.
# Spring: March–May 2022
spring_nc = nc[nc['appointment_month'].isin(['2022-03', '2022-04',
spring_summary = spring_nc.groupby(['appointment_month', 'service_s

plt.figure(figsize=(12, 6))
sns.lineplot(data=spring_summary, x='appointment_month', y='count_o

plt.title('Number of Appointments by Service Setting (March–May 202
plt.xlabel('Appointment Month')
plt.ylabel('Number of Appointments')
plt.xticks(rotation=45)
plt.legend(title='Service Setting')
plt.tight_layout()
plt.show()
```



Assignment Activity 5: Social Media Insights (X / Twitter)

We analyse NHS-related tweets to identify trending hashtags, public sentiment, and external insights that support service planning.

In [91]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set figure size and style
plt.rcParams['figure.figsize'] = (15, 12)
sns.set(style="white")

# Set maximum column width for readability
pd.options.display.max_colwidth = 200
```

In [92]:
```
# describe the the dataset
tweets.describe()
```

Out[92]:

	tweet_id	tweet_retweet_count	tweet_favorite_count
count	1.174000e+03	1174.000000	1174.00000
mean	1.567612e+18	8.629472	0.37138
std	2.427553e+13	29.784675	2.04470
min	1.567574e+18	0.000000	0.00000
25%	1.567590e+18	0.000000	0.00000
50%	1.567611e+18	1.000000	0.00000
75%	1.567633e+18	3.000000	0.00000
max	1.567655e+18	303.000000	42.00000

In [93]:
```
# show the info of the dataset
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1174 entries, 0 to 1173
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   tweet_id                1174 non-null   int64 
 1   tweet_full_text         1174 non-null   object
 2   tweet_entities          1174 non-null   object
 3   tweet_entities_hashtags 1007 non-null   object
 4   tweet_metadata          1174 non-null   object
 5   tweet_retweet_count     1174 non-null   int64 
 6   tweet_favorite_count    1174 non-null   int64 
 7   tweet_favorited         1174 non-null   bool  
 8   tweet_retweeted         1174 non-null   bool  
 9   tweet_lang              1174 non-null   object
dtypes: bool(2), int64(3), object(5)
memory usage: 75.8+ KB
```

In [94]:
```
# show the head of the dataset
```

```
tweets.head()
```

Out[94]:

	tweet_id	tweet_full_text	tweet_entities	twe
0	1567629223795527681	As Arkansas' first Comprehensive Stroke Certified Center, UAMS provides Arkansans with access to the most advanced stoke care. Join us in our mission to make a difference in the health and well-be...	{'hashtags': [{'text': 'Healthcare', 'indices': [253, 264]}], 'symbols': [], 'user_mentions': [], 'urls': [{'url': 'https://t.co/yw0cstfmSI', 'expanded_url': 'https://bit.ly/3BiSKbs', 'display_url...	
1	1567582846612553728	RT @AndreaGrammer: Work-life balance is at the foundation of how decisions are made and where #PremiseHealth is headed. We're #hiring for...	{'hashtags': [{'text': 'PremiseHealth', 'indices': [94, 108]}, {'text': 'hiring', 'indices': [127, 134]}], 'symbols': [], 'user_mentions': [{'screen_name': 'AndreaGrammer', 'name': 'Andrea Grammer...	#
2	1567582787070304256	RT @OntarioGreens: $10 billion can go a long way to fixing our broken #Healthcare system.\n\nYet Doug Ford would rather spend it ALL on a hig...	{'hashtags': [{'text': 'Healthcare', 'indices': [70, 81]}], 'symbols': [], 'user_mentions': [{'screen_name': 'OntarioGreens', 'name': 'Green Party of Ontario', 'id': 37115912, 'id_str': '37115912'...	
3	1567582767625428992	RT @modrnhealthcr: 🚨 #NEW: 🚨 Insurance companies are figuring out the best ways to collect information about members' race and ethnicity data...	{'hashtags': [{'text': 'NEW', 'indices': [20, 24]}], 'symbols': [], 'user_mentions': [{'screen_name': 'modrnhealthcr', 'name': 'Modern Healthcare', 'id': 18935711, 'id_str': '18935711', 'indices':...	
4	1567582720460570625	ICYMI: Our recent blogs on Cybersecurity in Accounting https://t.co/4nnK0FiVVL and Digital Transformation in Healthcare Finance https://t.co/jlqn52lHD3 are a great read, take a look!\n\n#blogs #di...	{'hashtags': [{'text': 'blogs', 'indices': [184, 190]}, {'text': 'digitaltransformation', 'indices': [191, 213]}, {'text': 'cybersecurity', 'indices': [214, 228]}, {'text': 'accounting', 'indices'...	

In [95]:

```
# show the count of column tweet_retweet_count
tweets['tweet_retweet_count'].value_counts()
```

Out[95]:
```
tweet_retweet_count
0      526
1      215
2      114
3       70
5       35
4       27
7       18
12      16
8       15
73      14
9       13
6       12
208     12
35      10
37       6
11       6
10       5
53       5
44       4
150      4
63       4
76       3
85       3
41       3
62       3
207      3
68       3
78       2
23       2
24       2
72       2
16       2
13       1
49       1
48       1
15       1
107      1
14       1
79       1
20       1
39       1
19       1
303      1
57       1
40       1
54       1
169      1
Name: count, dtype: int64
```

In [96]:
```
# show the count of column tweet_favorite_count
tweets['tweet_favorite_count'].value_counts()
```

Out[96]:
```
tweet_favorite_count
0     1027
1       91
2       16
3       13
4        7
5        5
6        2
17       1
12       1
10       1
8        1
13       1
11       1
7        1
20       1
28       1
14       1
18       1
9        1
42       1
Name: count, dtype: int64
```

In [97]:
```
# create new dataframe with only the object (string) datatypes
tweets_text = tweets[['tweet_full_text', 'tweet_entities', \
                      'tweet_entities_hashtags', 'tweet_metadata', '
                      'tweet_lang']]

tweets_text.head()
```

Out[97]:

	tweet_full_text	tweet_entities	tweet_entities_hashtags	
0	As Arkansas' first Comprehensive Stroke Certified Center, UAMS provides Arkansans with access to the most advanced stoke care. Join us in our mission to make a difference in the health and well-be...	{'hashtags': [{'text': 'Healthcare', 'indices': [253, 264]}], 'symbols': [], 'user_mentions': [], 'urls': [{'url': 'https://t.co/yw0cstfmSI', 'expanded_url': 'https://bit.ly/3BiSKbs', 'display_url...	#Healthcare	{'
1	RT @AndreaGrammer: Work-life balance is at the foundation of how decisions are made and where #PremiseHealth is headed. We're #hiring for...	{'hashtags': [{'text': 'PremiseHealth', 'indices': [94, 108]}, {'text': 'hiring', 'indices': [127, 134]}], 'symbols': [], 'user_mentions': [{'screen_name': 'AndreaGrammer', 'name': 'Andrea Grammer...	#PremiseHealth, #hiring	{'
2	RT @OntarioGreens: $10 billion can go a long way to fixing our broken #Healthcare system.\n\nYet Doug Ford would rather spend it ALL on a hig...	{'hashtags': [{'text': 'Healthcare', 'indices': [70, 81]}], 'symbols': [], 'user_mentions': [{'screen_name': 'OntarioGreens', 'name': 'Green Party of Ontario', 'id': 37115912, 'id_str': '37115912'...	#Healthcare	{'
3	RT @modrnhealthcr: 🚨 #NEW: 🚨 Insurance companies are figuring out the best ways to collect information about members' race and ethnicity data...	{'hashtags': [{'text': 'NEW', 'indices': [20, 24]}], 'symbols': [], 'user_mentions': [{'screen_name': 'modrnhealthcr', 'name': 'Modern Healthcare', 'id': 18935711, 'id_str': '18935711', 'indices':...	#NEW	{'
4	ICYMI: Our recent blogs on Cybersecurity in Accounting https://t.co/4nnK0FiVVL and Digital Transformation in Healthcare Finance https://t.co/jlqn52lHD3 are a great read, take a look!\n\n#blogs #di...	{'hashtags': [{'text': 'blogs', 'indices': [184, 190]}, {'text': 'digitaltransformation', 'indices': [191, 213]}, {'text': 'cybersecurity', 'indices': [214, 228]}, {'text': 'accounting', 'indices'...	#blogs, #digitaltransformation, #cybersecurity, #accounting, #finance, #healthcare	{'

In [98]:
```
# split the hashtags from the tweet_full_text column
# and store in the tags variable
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
```

Out[98]:
```
['#healthcare',
 '#premisehealth',
 '#hiring',
 '#healthcare',
 '🚨#new:🚨',
 'look!\n\n#blogs',
 '#digitaltransformation',
 '#cybersecurity',
 '#accounting',
 '#finance',
 '#healthcare',
 'https://t.co/jrgqeqdme4\n.\n#firstcoastcna',
 '#cnaexam',
 '#cnaexampreparation',
 '#jacksonville',
 '#cnatraining',
 '#nurse',
 '#nursing',
 '#nurselife',
 '#nursepractitioner',
 '#nurseproblems',
 '#nursingschool',
 '#healthcare',
 '🚨#new:🚨',
 '#disparities.',
 '@karahartnett\n#healthcare',
 '#alert',
 '#insurance',
 '#data\nhttps://t.co/h9hlamr7p9',
 '#healthcare',
 '#healthcare',
 '❤️\n\n#healthcare',
 '#healthcare',
 '#hcldr',
 '#premisehealth',
 '#hiring',
 '#premisehealth',
 '#hiring',
 '#healthcare',
 '#qualitypatientcare',
 '#jobs',
 '#job',
 '#ascp2022',
 '#ascp100.',
 '#healthcare',
 '#healthsecretary',
 '#healthcare',
 '#ai,',
```

```
'#sdoh,',
'#healthcare',
'\nhttps://t.co/ewe7jntj1e\n#tropicana',
'#real',
'#juice',
'#healthcare',
'#watch',
'#worms',
'#fruits',
'#healthtips',
'#tips',
'#healthcare',
'#thewoodlands,',
'#healthcare',
'#chicago,',
'#healthcare',
'#telehealth',
'#healthcare',
'#virtualcare',
'#mediqueststaffing',
'#hospital',
'#shift',
'(#newportbeach)',
'#job',
'#mediqueststaffing',
'#hospital',
'#shift',
'(#newportbeach)',
'#job',
'#jobs',
'#employment',
'#healthcare',
'#job',
'#hiring',
'#antiracist',
'#healthcare:',
'#newhealthcare',
'#healthcare',
'#hospitals',
'#physicians',
'of\xa0#healthcare\xa0delivery:',
'#newhealthcare\xa0',
'#healthcare',
'#cloud',
'#ehr',
'#datasecurity',
'#options',
'#optionstrading',
'#pfizer',
'#optionsflow',
'#stocks',
'#stock',
'#stockmarket',
'#investing',
'#investment',
'#healthcare',
```

```
'#healthcare',
'#womenleaders',
'#healthcare',
'#albuquerque,',
'#healthcare',
'#healthcare',
'future\n\nhttps://t.co/hctoc1vady\n\n#telehealth',
'#telemedicine',
'#healthcare',
'#therapy',
'#primarycare',
'#home',
'#hit',
'#covid19',
'#usa',
'#selfcare',
'#coronavirus',
'#photooftheday',
'#healthcare.',
'#health.',
'#internalmedicine',
'#tipsfornewdocs',
'#meded',
'#medtwitter',
'#medicine',
'#medical',
'#medica…',
'#healthcare',
'#covid!',
'#stra…',
'#strategy',
'#competitiveintellige…',
'#strate…',
'#strategy',
'#competi…',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#healthcare',
'#biotech',
'#pharma',
'#competitivemarketing',
'#pharmaceutical',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#pharmaceutical',
'#competitivemarketing',
'#biotech',
'#healthcare',
'#pharma',
'#internalmedicine',
'#tipsfornewdocs',
'#meded',
'#medtwitter',
'#medicine',
```

```
'#medical',
'#medica…',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#pharmaceutical',
'#biotech',
'#competitivemarketing',
'#pharma',
'#healthcare',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#pharmaceutical',
'#pharma',
'#biotech',
'#competitivemarketing',
'#healthcare',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#healthcare',
'#pharmaceutical',
'#competitivemarketing',
'#pharma',
'#biotech',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#healthcare',
'#biotech',
'#competitivemarketing',
'#pharma',
'#pharmaceutical',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#pharma',
'#healthcare',
'#biotech',
'#competitivemarketing',
'#pharmaceutical',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#pharma',
'#competitivemarketing',
'#pharmaceutical',
'#healthcare',
'#biotech',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#healthcare',
'#biotech',
'#pharma',
```

```
'#productmarketing',
'#pharmaceutical',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#productmarketing',
'#healthcare',
'#pharma',
'#biotech',
'#pharmaceutical',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#biotech',
'#pharmaceutical',
'#healthcare',
'#pharma',
'#productmarketing',
'#lenexa,',
'#healthcare',
'#healthcare',
'delivery.\nhttps://t.co/etq4tbb6vu\n#hippotechnologies',
'#virtualcare',
'#headset',
'#wearabletech',
'#wearables',
'#healthcare',
'#firstresponder',
'#teletriage',
'#worldhealthorganization',
'#ascp2022',
'#ascp100.',
'#healthcare.',
'\n\n#healthcare',
'#revenuecycle',
'#healthit',
'#saskatchewan',
'https://t.co/xxahgpsvsz\n#healthcare',
'#business',
'#datasecurity',
'#compliance\ncloud',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#qua…',
'#vaccine',
'#covid',
```

```
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#qua…',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#qua…',
'#epic',
'#identitybridge…',
'#epic',
'#identitybridge…',
'#medicaid',
'#heartfailure',
'#barostim',
'@lluhealth!\n\n#outsmarttheheart',
'#cardiology',
'#healthcare',
'#medtech',
'#innovation',
'#familydoctor.',
'#globalaussieawards',
'#vr',
'#ar',
'#healthcare',
'#healthcare',
'itself\n\n#cdnpoli',
'#holyoke,',
'#healthcare',
'#healthcare',
'#mammogr…',
'#healthcare',
'#speechpathology',
'#healthcare',
'https://t.co/bmvwpsfker\n\n#fitness',
'#beuty',
'#healthcare',
'#fitnesslife',
'#beauty',
'#beautytips',
```

```
'#beaut',
'#healthcare',
'#birmingham,',
'#privacyofficer',
'#healthcare',
'#webinar',
'come!!\n\n#race4…',
'#healthcareisahumanright',
'#prep',
'#hiv',
'#healthcare',
'#bluecrew',
'#voteblue',
'#nursetwitter',
'#healthcare',
'#epic',
'#identitybridge',
'\n\n#itsecurity',
'#cybersecurity',
'#iam',
'#datasecurity',
'#healthcare',
'#hipaa',
'#healthcare',
'#prep',
'#healthcare',
'https://t.co/luvbnj8opf\n\n#healthcare',
'#innovation',
'#customerexperience',
'#healthcare',
'#healthcare',
'#security',
'#healthcare',
'#centralized',
'\n#humanrights',
'#healthcare',
'\n#truth',
'#truth',
'\n#nazi',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#qua…',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
```

```
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#qua…',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#quarantine',
'#influenza',
'#news',
'#doctor',
'#stayhome',
'#healthcare',
'#who',
'#flu',
'#immunization',
'#truth',
'#social-distancing',
'\n#detroit',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#qua…',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
```

```
'#lockdown',
'#science',
'#quarantine',
'#influenza',
'#news',
'#doctor',
'#stayhome',
'#healthcare',
'#who',
'#flu',
'#immunization',
'#truth',
'#social-distancing',
'\n#detroit',
'#vaccine',
'#covid',
'#coronavirus',
'#pandemic',
'#health',
'#corona',
'#vaccines',
'#vaccination',
'#virus',
'#medicine',
'#lockdown',
'#science',
'#quarantine',
'#influenza',
'#news',
'#doctor',
'#stayhome',
'#healthcare',
'#who',
'#flu',
'#immunization',
'#truth',
'#social-distancing',
'\n#detroit',
'#data',
'#healthcare?',
'#healthcare',
'#healthcare',
'#ontariohealthcare',
'#healthcare',
'#regulation',
'#itsm',
'#health',
'#healthcare',
'#tech…',
'#healthcare',
'#palliative',
'#care',
'#healthcare',
'#congenital',
'#cardiotwitter',
'#doctors',
```

```
'#hospital',
'#clinics',
'#health',
'#healthcareit',
'#ehealth',
'#healthcare',
'#healthcare',
'#ai',
'#healthcare',
'#technology',
'#soaak',
'#healthcare',
'#mentalhealth',
'#therapy',
'#anxiety',
'#stress',
'#emergingtech',
'#coruzant',
'#thedigitalexecutive',
'#dental',
'#practice',
'#patients',
'team.\nhttps://t.co/jt60d4ab4v\n#dentalcare',
'#dentistry',
'#healthtech',
'#healthcareit',
'#healthcare',
'#digitalhealth',
'#healthcare',
'#technology',
'#healthcare',
'#ontario',
'#generalstrike',
'#workersunite',
'#onpoli',
'#cdnpoli',
'#education',
'#healthcare',
'#healthcare',
'#chicago,',
'#specialneedsparenting',
'#healthcare',
'\n\n#skincare',
'#technology',
'#cosmetics',
'#beauty',
'#healt…',
'#healthcare',
'#healthcare',
'#ai',
'#cheverly,',
'#healthcare',
'\n#lifescience',
'#healthcare',
'#quantiphi',
'#ai-led',
```

```
 '#medicalimaging:',
 '\n#solvingwhatmatters',
 '#healthcare',
 '#aiinhealthcare',
 '#ai',
 '#healthcare',
 '#drugdiscovery\n@closedloopai',
 '#patientdata…',
 '#monkeypox',
 '\n#monkeypoxcases',
 '#monkeypoxvaccine',
 '#deathrate',
 '#vaccine',
 '#healthcare',
 '#monkeypox',
 '\n#monkeypoxcases',
 '#monkeypoxvaccine',
 '#deathrate',
 '#vaccine',
 '#healthcare',
 '#antiracist',
 '#healthcare:',
 '#örebro',
 '#healthcare',
 '#career',
 '—#transportation/#construction/#telecommunications/#lawenforceme
nt/#healthcare',
 '#veterans',
 '#healthcare',
 '#mammogram',
 '#mobilehealth',
 '#healthsecutity',
 '#breastcancerawareness',
 '#ruralhealth',
 '#healthcare',
 '\n#medicine',
 '#doctor',
 '#healthcare…',
 '#primarycare',
 '#healthcare',
 '#maryland',
 '#healthcare,',
 '#healthcare',
 '#covid19',
 'further👉https://t.co/zkx4vl4sxf\n\n#clubbing',
 '#nail',
 '#diseases',
 '#health',
 '#healthcare',
 '#healthylifestyle',
 '\n\nhttps://t.co/n8mboeqnbk\n\n#healthcare',
 '#personalfinance',
 '#medical',
 '#healthcare…',
 '#ai',
 '#healthcare',
```

```
'#drugdiscovery\n@closedloopai',
'#patientdata\n@aitherightway',
'#eyedisease\n@cleerly_inc',
'#imaging\n@owkinscience',
'#personalizedmedicine\n@deepcell_inc',
'#cellbiology',
'#research',
'#memphis:',
'#media',
'#socialmedia',
'#healthcare',
'#business.',
'firm.\n\n#healthcare',
'#medical',
'#healthcare',
'#n95',
'#masks',
'#cdnpoli',
'#healthcare',
'#goals',
'#potential.',
'#nursingcare',
'#linkinbio',
'#nursing',
'#ohio',
'#health',
'#healthcare.',
'planting...\n\n#appleevent',
'#iphone',
'#ems',
'#healthcare',
'#genzeon',
'#healthcare',
'#automation',
't-square\n#financialastrology',
'#gold',
'#nasdaq',
'#hea…',
'#healthcare',
'#healthcare',
'#chicago,',
'#substanceabuse',
'#healthcare',
'#healthcare.',
'needs.\n\n#appleevent',
'#match',
'#healthcare',
'#education',
'#usa',
'#ruralhealth',
'system!#match2023',
'#match2023madness',
'#usmle',
'#img',
'#cloud',
'#healthcare',
```

```
'#healthcare',
'#chicago,',
'#healthcare',
'#saskatchewan',
'#ai',
'#healthcare!\n\ninfographic',
'@ingliguori\n\n#python',
'#nodejs',
'#machinelearning',
'#100daysofmlcode',
'#html5',
'#rstats',
'#s…',
'#healthcare',
'#losangeles,',
'\n\n#comedy',
'#adulting',
'#healthcare',
'#esmo22:',
'#esmo22:',
'#ai',
'#healthcare!\n\ninfographic',
'@ingliguori\n\n#python',
'#nodejs',
'#machinelearning',
'#100daysofmlcode',
'#html5',
'#rstats',
'#s…',
'#esmo22:',
'#ai',
'#healthcare!\n\ninfographic',
'@ingliguori\n\n#python',
'#nodejs',
'#machinelearning',
'#100daysofmlcode',
'#html5',
'#rstats',
'#s…',
'#esmo22:',
'#ai',
'#healthcare!\n\ninfographic',
'@ingliguori\n\n#python',
'#nodejs',
'#machinelearning',
'#100daysofmlcode',
'#html5',
'#rstats',
'#s…',
'#esmo22:',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#healthcare',
'#pharmaceutical',
'#productmarketing',
```

```
'#biotech',
'#pharma',
'#esmo22:',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#pharma',
'#biotech',
'#productmarketing',
'#healthcare',
'#pharmaceutical',
'#esmo22:',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#healthcare',
'#pharmaceutical',
'#productmarketing',
'#biotech',
'#pharma',
'#esmo22:',
'#strategy',
'#competitiveintelligence',
'#marketing',
'#productmarketing',
'#biotech',
'#healthcare',
'#pharma',
'#pharmaceutical',
'#premisehealth',
'#hiring',
'#healthcare',
'#qualitypatientcare',
'#jobs',
'#job',
'#healthcare',
'#doctors',
'#mediqueststaffing',
'#medical',
'#assistant!',
'(#suncity)',
'#job',
'#jobs',
'#employment',
'#healthcare',
'#job',
'#hiring',
'#mediqueststaffing',
'#jobsearch',
'#portolavalley,',
'#healthcare',
'open\n\n#gvwire',
'#news',
'#politics',
'#healthcare',
'#economy',
```

```
'#tech',
'#califor…',
'initiatives\n\n#healthcare',
'#mentalhealth',
'#pandemic',
'#capitalism',
'us.\n\n#infrastructure',
'\n#climateemergency',
'\n#utilities',
'\n#foodinsecurity\n#water\n#healthcare',
'#buffalo,',
'#healthcare',
'#womeninmedicine',
'#healthcare',
'#womeninmedicine',
'#healthcare',
'#patientexperience',
'#tech',
'#healthcareprofessionals',
'#5g',
'#healthcare.',
'#healthtech',
'#healthcare',
'#medtech',
'#internalmedicine',
'#tipsfornewdocs',
'#meded',
'#medtwitter',
'#medicine',
'#medical',
'#medica…',
'#hospitalstalktolovedones',
'#education',
'#ptsafet…',
'#brexit',
'#eu',
'🌟\n\n👉👉https://t.co/de14oseogo\n\n#indianbusinesslisting',
'#localsearchlisting',
'#localseo',
'#healthcare',
'#lifestyle',
'#theclairvoyantregister',
'#dating',
'#theadultentertainment',
'#healthcare',
'#chicago,',
'#artificialintelligence’s',
'#healthcare\nby',
'https://t.co/rro0ioptmm\n\n#ai',
'#iot',
'#bigdata',
'#machinelearning\n\ncc:',
'\nhttps://t.co/4h1wip7jbs\n\n#centurycollege',
'#nursingassistant',
'#nursingassistanttraining',
'#na',
```

```
'#nahha',
'#hha',
'#mnnursingassistant',
'#healthcare',
'#healthcarecareers',
'#obra',
'#online',
'#hybridlearning',
'#startups',
'#leaders',
'#investors.\nhttps://t.co/ck3hgpimt4\n#innovations',
'#femtech',
'#womenentrepreneurs',
'#healthcare',
'#robotics',
'#5g',
'#iot',
'#healthcare.',
'#mobileff',
'#palmcoast,',
'#healthcare',
'information\n#healthcare',
'#health',
'\n#santshriasharamjibapu',
'#hiv/#aids',
'“#christian”',
'#universal',
'#healthcare',
'#healthcare?',
'#hospitalstalktolovedones',
'#education',
'#ptsafet…',
'#digitalhealth!\n—',
'#medtech',
'#hcldr',
'#healthcare',
'(#matthews,',
'#job',
'#doctorofveterinarymedicine',
'#doctorofveterinarymedicine',
'#preventivecare',
'#formularymanagement',
'#medicalrecords',
'#hospitaloperations',
'#wellness',
'#healthcare',
'#patientcare',
'death.\n\n#healthfacts',
'#healthylifestyletips',
'#healthcare',
'#toomuchsleep',
'#sittingandhealth',
'#sittingtoomuch',
'#digitaltransformation',
'#healthcare.\n\n#healthtech',
'#iot',
```

```
'#dataanalytics',
'#data',
'#medtech',
'#tech',
'#digitaltransformation',
'#healthcare?',
'#administration',
'#heal…',
'#healthcare',
'#education.',
'(#rockhill,',
'#job',
'#doctorofveterinarymedicine',
'#doctorofveterinarymedicine',
'#preventivecare',
'#formularymanagement',
'#medicalrecords',
'#hospitaloperations',
'#wellness',
'#healthcare',
'#patientcare',
'#cloud',
'#healthcare',
'#cloud',
'#healthcare',
'#healthcare',
'#patientcare',
'#d2o',
'#healthcare',
'#artificialintelligence',
'#supplychain',
'#monkeypox',
'\n#monkeypoxcases',
'#monkeypoxvaccine',
'#deathrate',
'#vaccine',
'#healthcare',
'#monkeypox',
'\n#monkeypoxcases',
'#monkeypoxvaccine',
'#deathrate',
'#vaccine',
'#healthcare',
'(#charlotte,',
'#job',
'#doctorofveterinarymedicine',
'#doctorofveterinarymedicine',
'#preventivecare',
'#formularymanagement',
'#medicalrecords',
'#hospitaloperations',
'#wellness',
'#healthcare',
'#patientcare',
'#healthcare',
'#hiv/#aids',
```

```
'“#christian”',
'#universal',
'#healthcare',
'#faithregional',
'#healthcare',
'#healthcare',
'#worcesterma',
'(#winstonsalem,',
'#job',
'#doctorofveterinarymedicine',
'#doctorofveterinarymedicine',
'#preventivecare',
'#formularymanagement',
'#medicalrecords',
'#hospitaloperations',
'#wellness',
'#healthcare',
'#patientcare',
'#hospitalstalktolovedones',
'#education',
'#ptsafet…',
'#new:',
'#coronavirusvaccines',
'to:\nhttps://t.co/wukk9hk1vz\n\n#hospital',
'#healthcare',
'#catholic',
'#catholichospitals',
'#hospitalstalktolovedones',
'#education',
'#ptsafet…',
'implementation.\n#healthcare',
'#trangendercare',
'#genderaffirmingcare',
'#…',
'(#wesleychapel,',
'#job',
'#doctorofveterinarymedicine',
'#doctorofveterinarymedicine',
'#preventivecare',
'#formularymanagement',
'#medicalrecords',
'#hospitaloperations',
'#wellness',
'#healthcare',
'#patientcare',
'#healthcare?',
'#hiv/#aids',
'“#christian”',
'#universal',
'#healthcare',
'#medicareforall',
'#voteblue',
'#votebluetosavedemocracy',
'#healthcare',
'#mutualaidrequest?',
'#healthcare',
```

```
 '#healthcare',
 ...]
```

In [99]:
```
# turn the tags variable into a pandas series
series_tags = pd.Series(tags)

# show the head of the pandas series
series_tags.head()
```

Out[99]:
```
0       #healthcare
1    #premisehealth
2           #hiring
3       #healthcare
4           🚨#new:🚨
dtype: object
```

In [100…
```
# aggregate the tags using value_counts to get the count of each ha
series_tags = pd.Series(tags)
data = series_tags.value_counts().reset_index()

# show the head
data.head(30)
```

Out[100…

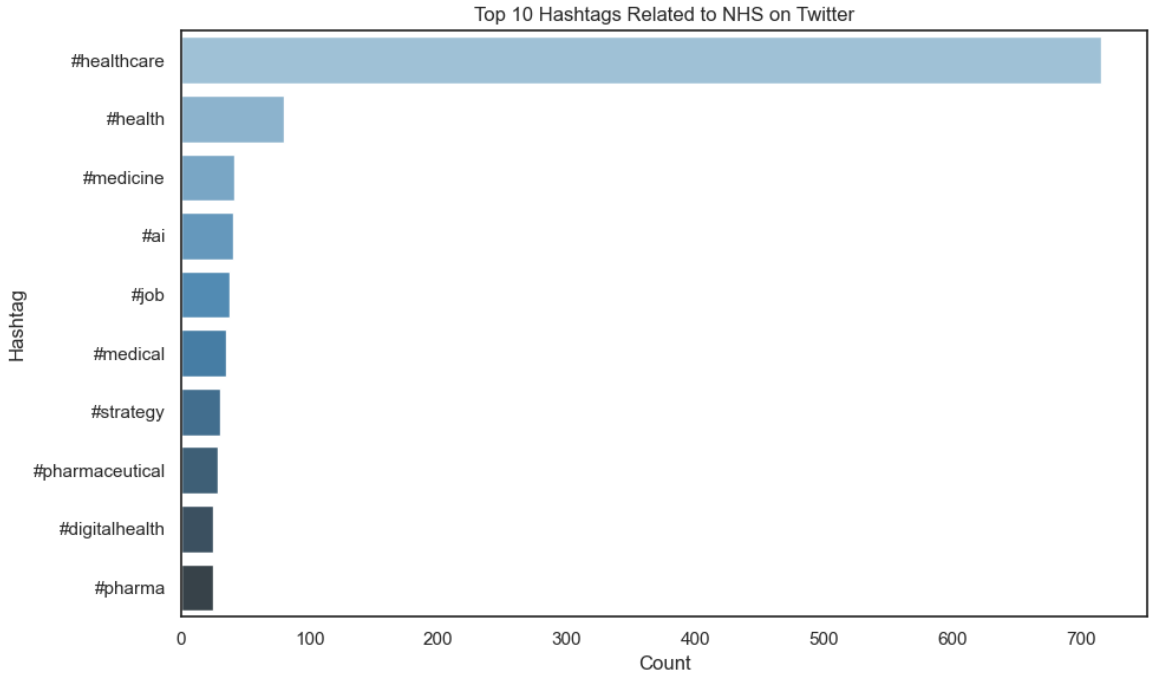
	index	count
0	#healthcare	716
1	#health	80
2	#medicine	41
3	#ai	40
4	#job	38
5	#medical	35
6	#strategy	30
7	#pharmaceutical	28
8	#digitalhealth	25
9	#pharma	25
10	#marketing	25
11	#medtwitter	24
12	#biotech	24
13	#competitiveintelligence	24
14	#meded	23
15	#vaccine	18
16	#hiring	18
17	#news	17
18	#machinelearning	17
19	#technology	17
20	#coronavirus	16
21	#womeninmedicine	16
22	#covid	16
23	#competitivemarketing	16
24	#wellness	15
25	#healthtech	15
26	#doctorofveterinarymedicine	14
27	#science	14
28	#medicare	14
29	#covid19	14

In [101…

```
data.columns = ['hashtag', 'count']
```

```
top_tags = data.head(10)

# Plot
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.barplot(data=top_tags, x='count', y='hashtag', palette='Blues_d
plt.title("Top 10 Hashtags Related to NHS on Twitter")
plt.xlabel("Count")
plt.ylabel("Hashtag")
plt.tight_layout()
plt.show()
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/384
0903982.py:9: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.
```



In [102…

```
# Convert to Series and count hashtags
series_tags = pd.Series(tags)
data = series_tags.value_counts().reset_index()
data.columns = ['hashtag', 'count']

# Show basic info and top 10
print(data.info())
display(data.head(10))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1754 entries, 0 to 1753
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   hashtag  1754 non-null   object
 1   count    1754 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 27.5+ KB
None
```

	hashtag	count
0	#healthcare	716
1	#health	80
2	#medicine	41
3	#ai	40
4	#job	38
5	#medical	35
6	#strategy	30
7	#pharmaceutical	28
8	#digitalhealth	25
9	#pharma	25

In [103…
```
# describe the new dataframe
data.describe()
```

Out[103…

	count
count	1754.000000
mean	2.470353
std	17.462844
min	1.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	716.000000

In [104…
```
# subset the dataframe to be count greater than 25
data_subset = data.loc[data['count'] > 25]
```

In [105…
```
# show the shape
data_subset.shape
```

Out[105… (8, 2)

In [106…
```
# Convert list to Series
tags_series = pd.Series(tags)

# Count hashtag frequency
top_tags = tags_series.value_counts()

# Display the top 30 hashtags
print("Top 30 Hashtags:")
print(top_tags.head(30))
```

```
Top 30 Hashtags:
#healthcare                    716
#health                         80
#medicine                       41
#ai                             40
#job                            38
#medical                        35
#strategy                       30
#pharmaceutical                 28
#digitalhealth                  25
#pharma                         25
#marketing                      25
#medtwitter                     24
#biotech                        24
#competitiveintelligence        24
#meded                          23
#vaccine                        18
#hiring                         18
#news                           17
#machinelearning                17
#technology                     17
#coronavirus                    16
#womeninmedicine                16
#covid                          16
#competitivemarketing           16
#wellness                       15
#healthtech                     15
#doctorofveterinarymedicine     14
#science                        14
#medicare                       14
#covid19                        14
Name: count, dtype: int64
```

In [107…
```
# Convert to DataFrame and reset index
top_tags_df = top_tags.reset_index()
top_tags_df.columns = ['hashtag', 'count']
top_tags_df['count'] = top_tags_df['count'].astype(int)

# View the cleaned hashtag count table
print(top_tags_df.head())
```

```
       hashtag  count
0  #healthcare    716
1      #health     80
2    #medicine     41
3          #ai     40
4         #job     38
```

Twitter Sentiment: Top NHS-related Hashtags

This chart displays the most frequently used hashtags from Twitter posts mentioning the NHS.

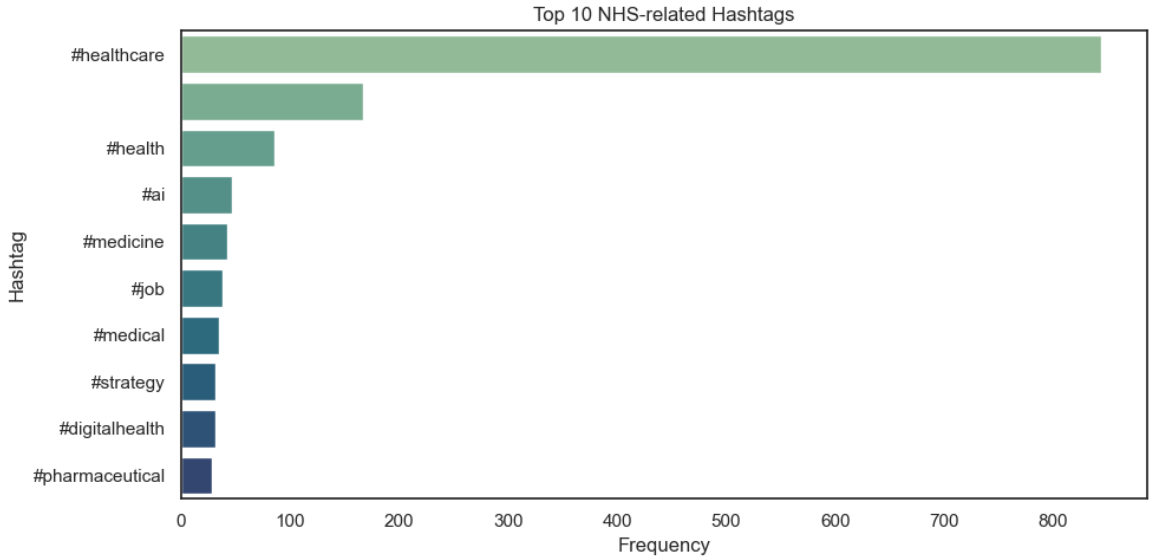
- **#Healthcare** dominates public conversation.
- Tags like **#AI** and **#DigitalHealth** show growing interest in tech-driven care.
- Other popular tags include **#MentalHealth**, **#Medicine**, and **#Pharmaceutical** — pointing to access, wellbeing, and service delivery concerns.

These findings reinforce the NHS's need to invest in digital infrastructure and understand patient sentiment in real time.

In [109…

```
# === Twitter Hashtag Frequency ===
tweets['tweet_entities_hashtags'] = tweets['tweet_entities_hashtags
hashtags = tweets['tweet_entities_hashtags'].str.split(',').explode
top_tags = hashtags.value_counts().head(10)

plt.figure(figsize=(10, 5))
sns.barplot(x=top_tags.values, y=top_tags.index, palette='crest')
plt.title('Top 10 NHS-related Hashtags')
plt.xlabel('Frequency')
plt.ylabel('Hashtag')
plt.tight_layout()
plt.show()
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/278
7226917.py:7: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.
```



In [184…
```
# Load Twitter dataset
import pandas as pd
from collections import Counter
import re

# Read the tweets file
tweets = pd.read_csv('tweets.csv')

# Extract hashtags using regular expressions
tweets['hashtags'] = tweets['tweet_full_text'].apply(lambda x: re.f

# Flatten and count hashtag frequency
hashtag_list = [tag for tags in tweets['hashtags'] for tag in tags
hashtag_freq = Counter(hashtag_list)
top_hashtags = hashtag_freq.most_common(10)
top_hashtags
```

Out[184…
```
[('#healthcare', 844),
 ('#health', 88),
 ('#ai', 47),
 ('#medicine', 42),
 ('#job', 38),
 ('#medical', 35),
 ('#strategy', 31),
 ('#digitalhealth', 31),
 ('#pharmaceutical', 28),
 ('#medtwitter', 27)]
```

Sentiment Analysis of NHS-related Tweets

Using TextBlob, we analysed the sentiment of NHS-related tweets. The results show:

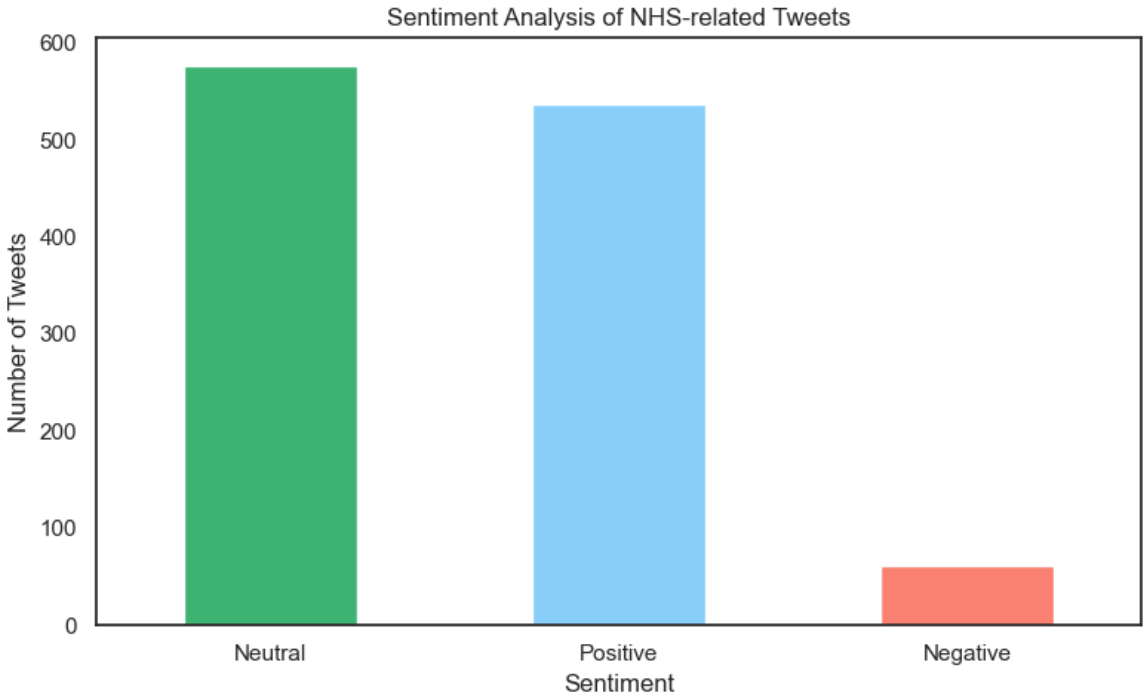
- A **majority of tweets are neutral or positive**, suggesting either objective news-sharing or general support for healthcare efforts.
- Only a **small fraction (~6%) of tweets are negative**, which may point to specific complaints or dissatisfaction with services.

- **Recommendation**: Regularly monitoring sentiment on platforms like X (Twitter) could help NHS communication teams detect public frustration early and tailor proactive outreach.

This analysis reinforces the value of **external, real-time data** to complement internal metrics on service delivery.

In [186…

```
# Step 1: Import libraries
from textblob import TextBlob
import pandas as pd
import matplotlib.pyplot as plt

# Step 2: Load and clean the tweets dataset
tweets = pd.read_csv('tweets.csv')
tweets = tweets.dropna(subset=["tweet_full_text"])  # Use the corre

# Step 3: Calculate sentiment polarity
tweets['polarity'] = tweets['tweet_full_text'].apply(lambda x: Textl

# Step 4: Classify sentiment
def classify_sentiment(score):
    if score > 0.1:
        return "Positive"
    elif score < -0.1:
        return "Negative"
    else:
        return "Neutral"

tweets['sentiment'] = tweets['polarity'].apply(classify_sentiment)

# Step 5: Count sentiment types
sentiment_counts = tweets['sentiment'].value_counts()

# Step 6: Plot the sentiment results
plt.figure(figsize=(8, 5))
sentiment_counts.plot(kind='bar', color=['mediumseagreen', 'lightsk
plt.title("Sentiment Analysis of NHS-related Tweets")
plt.xlabel("Sentiment")
plt.ylabel("Number of Tweets")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

# Optional: Print the count values
print(sentiment_counts)
```




```
sentiment
Neutral     576
Positive    537
Negative     61
Name: count, dtype: int64
```

Assignment activity 6

Make recommendations to the NHS.

In [188…

```python
# 1. Import required library
import pandas as pd

# 2. Load the dataset and assign to 'ar'
ar = pd.read_csv("appointments_regional.csv")

# 3. Convert the 'appointment_month' column to datetime
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'], e

# 4. Preview the first five rows
ar.head()

# 5. Print the minimum and maximum appointment dates
print("Minimum date in dataset:", ar['appointment_month'].min())
print("Maximum date in dataset:", ar['appointment_month'].max())

# 6. Filter the data to only include entries from 2021-08-01 onward
ar_filtered = ar[ar['appointment_month'] >= '2021-08-01'].copy()

# 7. View the filtered DataFrame
ar_filtered.head()
```

```
Minimum date in dataset: 2020-01-01 00:00:00
Maximum date in dataset: 2022-06-01 00:00:00
```

Out[188…

	icb_ons_code	appointment_month	appointment_status	hcp_type	ap
3652	E54000034	2021-08-01	Attended	GP	
3653	E54000034	2021-08-01	Attended	GP	
3654	E54000034	2021-08-01	Attended	GP	
3655	E54000034	2021-08-01	Attended	GP	
3656	E54000034	2021-08-01	Attended	GP	

Question 1: Should the NHS start looking at increasing staff levels?

NHS Monthly Capacity Utilisation

This analysis estimates how closely NHS appointment delivery aligns with national capacity limits.

- **Benchmark**: NHS capacity is assumed at **1.2 million appointments per weekday**.
- **Period Covered**: August 2021 to June 2022.
- **Method**:
  - Counted monthly appointments from actual data.
  - Calculated working days per month.
  - Estimated monthly capacity = Working Days × 1,200,000.
  - Calculated Utilisation (%) = (Actual Appointments / Estimated Capacity) × 100.

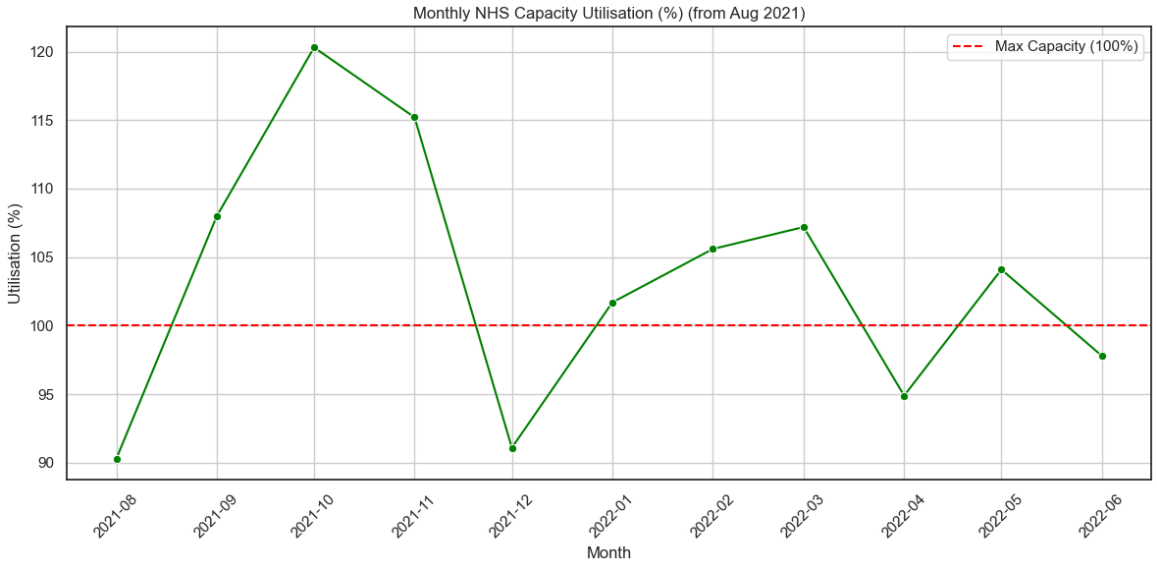
Key Insights:

- NHS exceeded its estimated capacity in several months, particularly:
  - **October 2021** (120.3%)
  - **November 2021** (115.2%)
- **December dips** below 92% likely reflect holiday impact.
- Sustained utilisation **above 100%** suggests ongoing pressure on NHS systems.
- A red dashed line at 100% on the chart marks the maximum planned capacity. Months crossing this line may indicate a need for **additional staffing**, **redistribution of appointment load**, or **policy adjustment**.

In [190…

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load the dataset
ar = pd.read_csv("appointments_regional.csv")
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'], e

# Filter data from August 2021 onward
ar_filtered = ar[ar['appointment_month'] >= '2021-08-01'].copy()

# Step 1: Group by appointment_month and sum appointments
ar_df = (
    ar_filtered.groupby('appointment_month')['count_of_appointments
    .sum()
    .reset_index()
)

# Step 2: Calculate working days for each month
ar_df['workdays'] = ar_df['appointment_month'].apply(
    lambda d: pd.date_range(start=d, end=d + pd.offsets.MonthEnd(0)
)

# Step 3: Estimate monthly capacity
DAILY_CAPACITY = 1_200_000
ar_df['estimated_capacity'] = ar_df['workdays'] * DAILY_CAPACITY

# Step 4: Calculate utilisation (%)
ar_df['utilisation'] = (ar_df['count_of_appointments'] / ar_df['est
ar_df['utilisation'] = ar_df['utilisation'].round(1)

# Step 5: Plot capacity utilisation over time (without lockdown sha
plt.figure(figsize=(12, 6))
sns.lineplot(data=ar_df, x='appointment_month', y='utilisation', ma
plt.axhline(y=100, linestyle='--', color='red', label='Max Capacity
plt.title('Monthly NHS Capacity Utilisation (%) (from Aug 2021)')
plt.xlabel('Month')
plt.ylabel('Utilisation (%)')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.legend()
plt.show()

# Optional: preview data
ar_df.head()
```

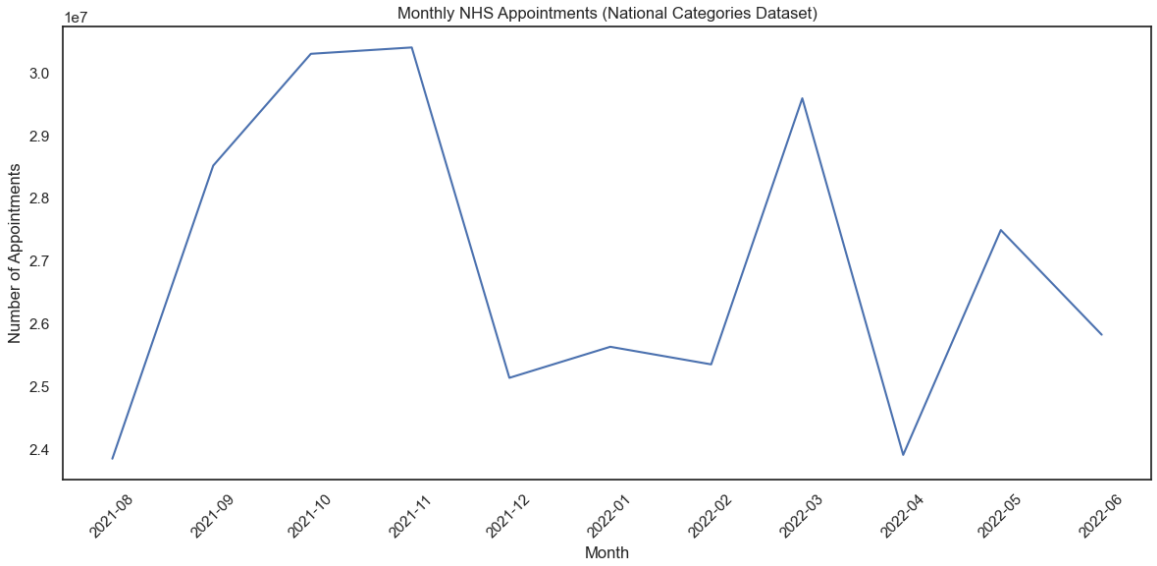


Out [190…

	appointment_month	count_of_appointments	workdays	estimated_capaci
0	2021-08-01	23852171	22	264000(
1	2021-09-01	28522501	22	264000(
2	2021-10-01	30303834	21	252000(
3	2021-11-01	30405070	22	264000(
4	2021-12-01	25140776	23	276000(

In [192…

```
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure date column is datetime
nc['appointment_month'] = pd.to_datetime(nc['appointment_month'], e

# Group and sum appointments per month
monthly_appointments_nc = nc.groupby(nc['appointment_month'].dt.to_
monthly_appointments_nc['appointment_month'] = monthly_appointments_

# Plot
plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_appointments_nc, x='appointment_month', y=
plt.title('Monthly NHS Appointments (National Categories Dataset)')
plt.xlabel('Month')
plt.ylabel('Number of Appointments')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [194…
```
# Aggregate by month, HCP type, appointment status, mode, and time
ar_agg = ar.groupby(['appointment_month', 'hcp_type', 'appointment_
                     'appointment_mode', 'time_between_book_and_app

# View the aggregated DataFrame
print(ar_agg.head())
```

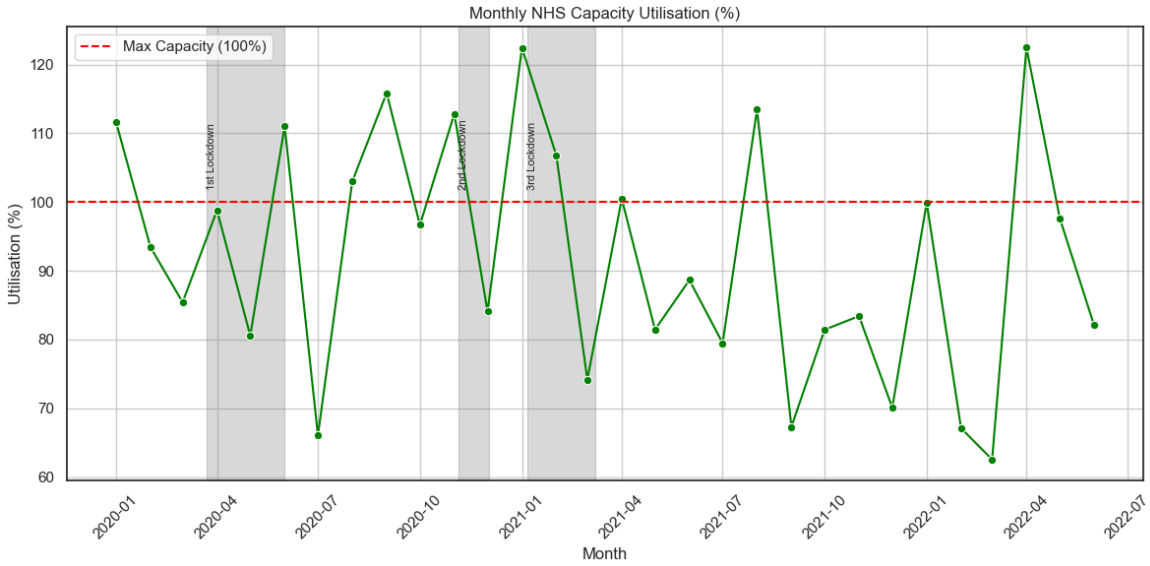
```
  appointment_month hcp_type appointment_status appointment_mode  \
0        2020-01-01       GP           Attended     Face-to-Face
1        2020-01-01       GP           Attended     Face-to-Face
2        2020-01-01       GP           Attended     Face-to-Face
3        2020-01-01       GP           Attended     Face-to-Face
4        2020-01-01       GP           Attended     Face-to-Face

  time_between_book_and_appointment  count_of_appointments
0                            1 Day                 653846
1                  15  to 21 Days                 620789
2                     2 to 7 Days                1723834
3                  22  to 28 Days                 421189
4                   8  to 14 Days                1123186
```

Figure: Monthly NHS Capacity Utilisation (%).

This line chart visualises monthly utilisation rates across the NHS between 2020 and mid-2022, using an estimated maximum daily capacity of 1.2 million appointments. The red dashed line marks 100% capacity, and shaded areas represent national COVID-19 lockdowns. Several months exceed 100%, indicating the NHS was operating beyond its assumed full capacity.

In [196…
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load or prepare your data (replace this with your actual dataset)
# ar_df = pd.read_csv("appointments_monthly.csv")  # if using a CSV
# For illustration purposes, we'll simulate
date_range = pd.date_range(start='2020-01-01', end='2022-06-01', fr
import numpy as np
np.random.seed(42)
```

```
appointment_counts = np.random.randint(16000000, 31000000, size=len

ar_df = pd.DataFrame({
    'appointment_month': date_range,
    'count_of_appointments': appointment_counts
})

# Calculate actual workdays per month
workdays = {
    month: pd.date_range(start=month, end=month + pd.offsets.MonthE
    for month in date_range
}
workdays_df = pd.DataFrame(list(workdays.items()), columns=['appoin

# Merge with your original data
merged_df = pd.merge(ar_df, workdays_df, on='appointment_month', ho

# Define daily capacity (e.g. 1.2 million appointments per working
DAILY_CAPACITY = 1_200_000
merged_df['estimated_capacity'] = merged_df['workdays'] * DAILY_CAP

# Calculate utilisation %
merged_df['utilisation'] = (merged_df['count_of_appointments'] / me

# Define lockdowns for annotation
lockdowns = [
    ('2020-03-23', '2020-06-01', '1st Lockdown'),
    ('2020-11-05', '2020-12-02', '2nd Lockdown'),
    ('2021-01-06', '2021-03-08', '3rd Lockdown')
]

# Create the plot
plt.figure(figsize=(12, 6))
sns.lineplot(x='appointment_month', y='utilisation', data=merged_df
plt.axhline(y=100, linestyle='--', color='red', label='Max Capacity

# Add shaded lockdown periods
for start, end, label in lockdowns:
    plt.axvspan(pd.to_datetime(start), pd.to_datetime(end), color='
    plt.text(pd.to_datetime(start), 102, label, rotation=90, fontsi

# Final plot settings
plt.title('Monthly NHS Capacity Utilisation (%)')
plt.xlabel('Month')
plt.ylabel('Utilisation (%)')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.legend()
plt.show()
```



In [198…

```
# Step 1: Load datasets
ar = pd.read_csv("appointments_regional.csv")
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'], e
icb_lookup = pd.read_csv("icb_locations.csv")

# Step 2: Filter appointments from August 2021 onwards
ar_filtered = ar[ar['appointment_month'] >= '2021-08-01'].copy()

# Step 3: Group total appointments per region and month
regional_monthly = (
    ar_filtered.groupby(['appointment_month', 'icb_ons_code'])['cou
    .sum()
    .reset_index()
)

# Step 4: Rename population column in icb_lookup
icb_lookup = icb_lookup.rename(columns={'icb_registered_population_

# Step 5: Merge population with appointments
regional_data = regional_monthly.merge(icb_lookup[['icb_ons_code',

# Step 6: Calculate workdays per month
regional_data['workdays'] = regional_data['appointment_month'].appl
    lambda d: pd.date_range(start=d, end=pd.to_datetime(d) + pd.off
)

# Step 7: Define national NHS daily capacity
NHS_DAILY_CAPACITY = 1_200_000
regional_data['monthly_capacity'] = regional_data['workdays'] * NHS_

# Step 8: Proportional regional capacity (based on population share
total_population = regional_data['population'].sum()
regional_data['regional_capacity'] = (regional_data['population'] /

# Step 9: Calculate utilisation percentage
regional_data['utilisation_pct'] = (regional_data['count_of_appointm
regional_data['utilisation_pct'] = regional_data['utilisation_pct']

# Step 10: Preview result
```

```
regional_data.head()
```

Out [198…

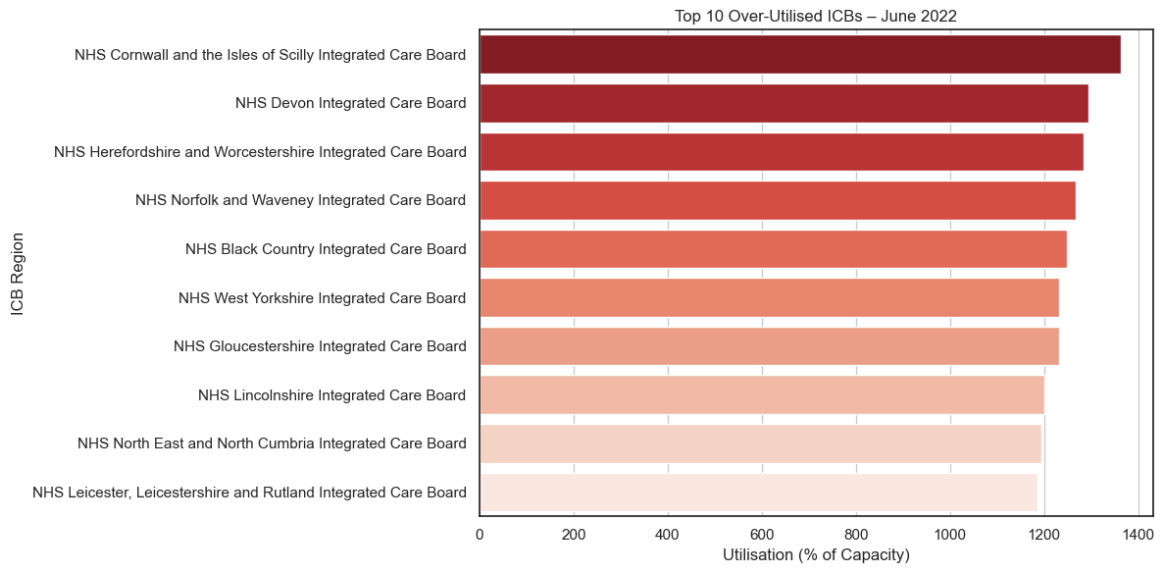
	appointment_month	icb_ons_code	count_of_appointments	population	v
0	2021-08-01	E54000008	1064804	2702059	
1	2021-08-01	E54000010	429720	1166953	
2	2021-08-01	E54000011	209824	516452	
3	2021-08-01	E54000013	332627	801457	
4	2021-08-01	E54000015	507015	1175420	

In [200…

```
# Step 1: Load datasets
ar = pd.read_csv("appointments_regional.csv")
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'], e
icb_lookup = pd.read_csv("icb_locations.csv")

# Step 2: Filter from August 2021 onwards
ar_filtered = ar[ar['appointment_month'] >= '2021-08-01'].copy()

# Step 3: Group monthly appointments by ICB
regional_monthly = (
    ar_filtered.groupby(['appointment_month', 'icb_ons_code'])['cou
    .sum()
    .reset_index()
)

# Step 4: Rename population column
icb_lookup = icb_lookup.rename(columns={'icb_registered_population_

# Step 5: Merge population into monthly appointments
regional_data = regional_monthly.merge(icb_lookup[['icb_ons_code',

# Step 6: Calculate monthly working days
regional_data['workdays'] = regional_data['appointment_month'].appl
    lambda d: pd.date_range(start=d, end=pd.to_datetime(d) + pd.off
)

# Step 7: Calculate national and regional capacity
NHS_DAILY_CAPACITY = 1_200_000
regional_data['monthly_capacity'] = regional_data['workdays'] * NHS_
total_population = regional_data['population'].sum()
regional_data['regional_capacity'] = (regional_data['population'] /

# Step 8: Calculate utilisation %
regional_data['utilisation_pct'] = (regional_data['count_of_appointm
regional_data['utilisation_pct'] = regional_data['utilisation_pct']

# Step 9: Merge in region names
regional_data_named = regional_data.merge(
    icb_lookup[['icb_ons_code', 'icb_ons_name']],
    on='icb_ons_code',
    how='left'
)
```

```
# Step 10: Clean column order
regional_data_named = regional_data_named[
    ['appointment_month', 'icb_ons_code', 'icb_ons_name',
     'count_of_appointments', 'population', 'workdays',
     'regional_capacity', 'utilisation_pct']
]

# Step 11: Visualise top 10 most over-utilised ICBs (latest month o
latest_month = regional_data_named['appointment_month'].max()
top10 = (regional_data_named[regional_data_named['appointment_month
         .sort_values(by='utilisation_pct', ascending=False)
         .head(10))

plt.figure(figsize=(12, 6))
sns.barplot(data=top10, x='utilisation_pct', y='icb_ons_name', pale
plt.title(f"Top 10 Over-Utilised ICBs – {latest_month.strftime('%B
plt.xlabel("Utilisation (% of Capacity)")
plt.ylabel("ICB Region")
plt.tight_layout()
plt.grid(True, axis='x')
plt.show()
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/222
0864005.py:58: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.
```

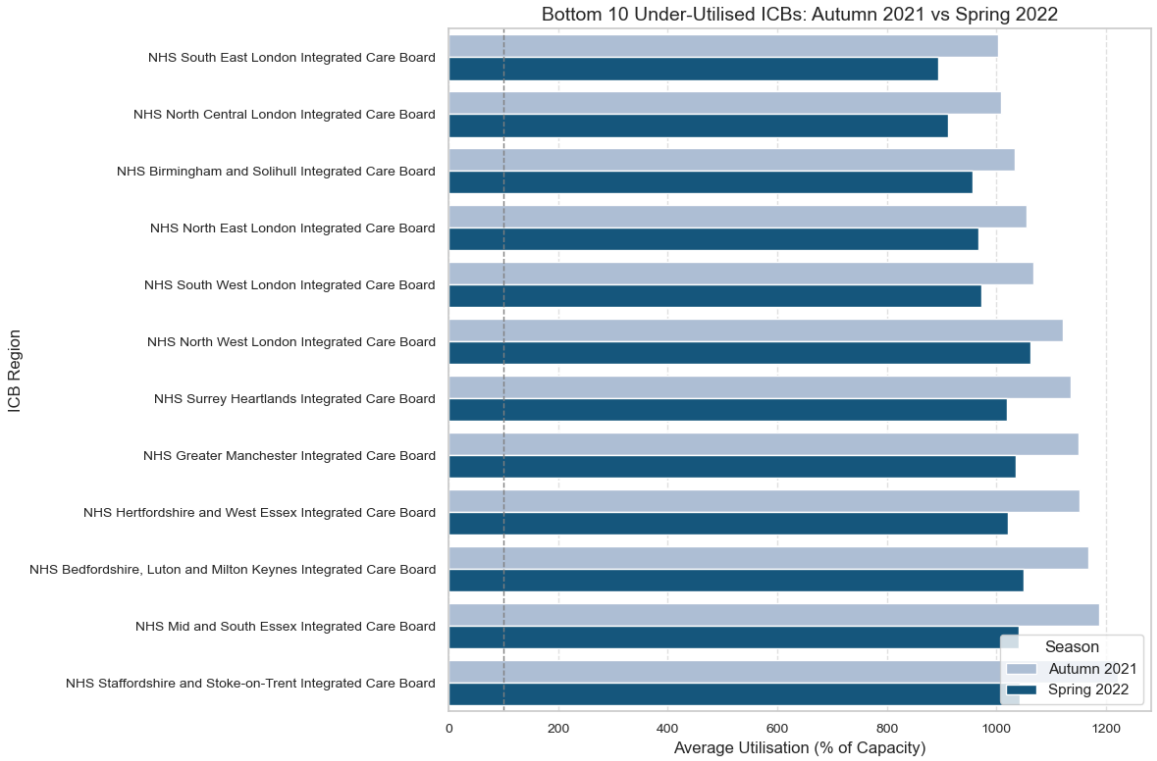


In [202…

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime

# Define Autumn and Spring months
```

```
autumn_months = [datetime.datetime(2021, 9, 1), datetime.datetime(2(
spring_months = [datetime.datetime(2022, 3, 1), datetime.datetime(2(

# Calculate average utilisation per ICB
autumn_under = regional_data_named[regional_data_named['appointment_
    .groupby('icb_ons_name')['utilisation_pct'].mean().reset_index(

spring_under = regional_data_named[regional_data_named['appointment_
    .groupby('icb_ons_name')['utilisation_pct'].mean().reset_index(

# Combine and find bottom 10 by utilisation from either season
under_combined = pd.concat([autumn_under, spring_under])
bottom_autumn = autumn_under.nsmallest(10, 'utilisation_pct')['icb_(
bottom_spring = spring_under.nsmallest(10, 'utilisation_pct')['icb_(
bottom_combined = pd.unique(bottom_autumn.tolist() + bottom_spring.t

# Filter to bottom ICBs only
bottom_df = under_combined[under_combined['icb_ons_name'].isin(bott(

# Reorder ICBs by Autumn 2021 values for consistent layout
order_icbs = autumn_under[autumn_under['icb_ons_name'].isin(bottom_(
    .sort_values(by='utilisation_pct')['icb_ons_name']

# Plot side-by-side bar chart
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")

ax = sns.barplot(
    data=bottom_df,
    y='icb_ons_name',
    x='utilisation_pct',
    hue='season',
    palette={'Autumn 2021': '#a6bddb', 'Spring 2022': '#045a8d'},
    order=order_icbs
)

plt.title("Bottom 10 Under-Utilised ICBs: Autumn 2021 vs Spring 202
plt.xlabel("Average Utilisation (% of Capacity)", fontsize=12)
plt.ylabel("ICB Region", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.axvline(100, linestyle='--', color='gray', linewidth=1)

plt.legend(title="Season", loc='lower right')
plt.tight_layout()
plt.show()
```

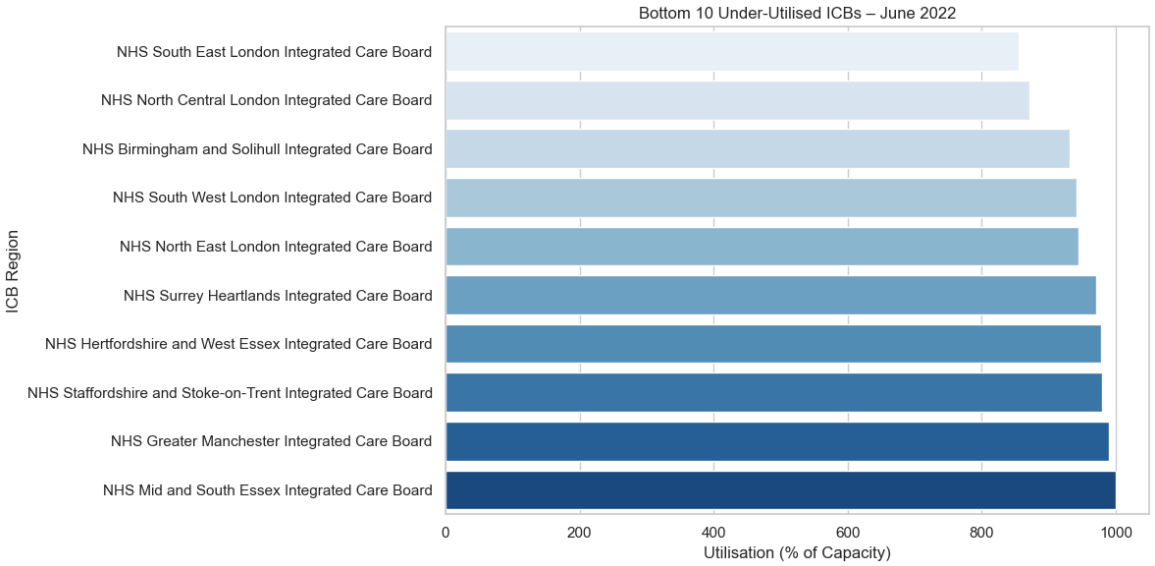
```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/297
2494017.py:21: FutureWarning:

unique with argument that is not not a Series, Index, ExtensionArra
y, or np.ndarray is deprecated and will raise in a future version.
```



In [204…

```
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Identify the most recent month in your data
latest_month = regional_data_named['appointment_month'].max()

# Step 2: Extract the bottom 10 ICBs by utilisation
bottom10 = (
    regional_data_named[regional_data_named['appointment_month'] ==
    .sort_values(by='utilisation_pct', ascending=True)
    .head(10)
)

# Step 3: Plot the bottom 10 under-utilised ICBs
plt.figure(figsize=(12, 6))
sns.barplot(data=bottom10, x='utilisation_pct', y='icb_ons_name', pa
plt.title(f"Bottom 10 Under-Utilised ICBs – {latest_month.strftime(
plt.xlabel("Utilisation (% of Capacity)")
plt.ylabel("ICB Region")
plt.tight_layout()
plt.grid(True, axis='x')
plt.show()
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/103
9882242.py:16: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.
```



Question 2: How Do Healthcare Professional Types Differ Over Time?

Appointment Trends by Healthcare Professional Type (Aug 2021–Jun 2022)

This chart shows the monthly volume of NHS appointments by type of healthcare professional.

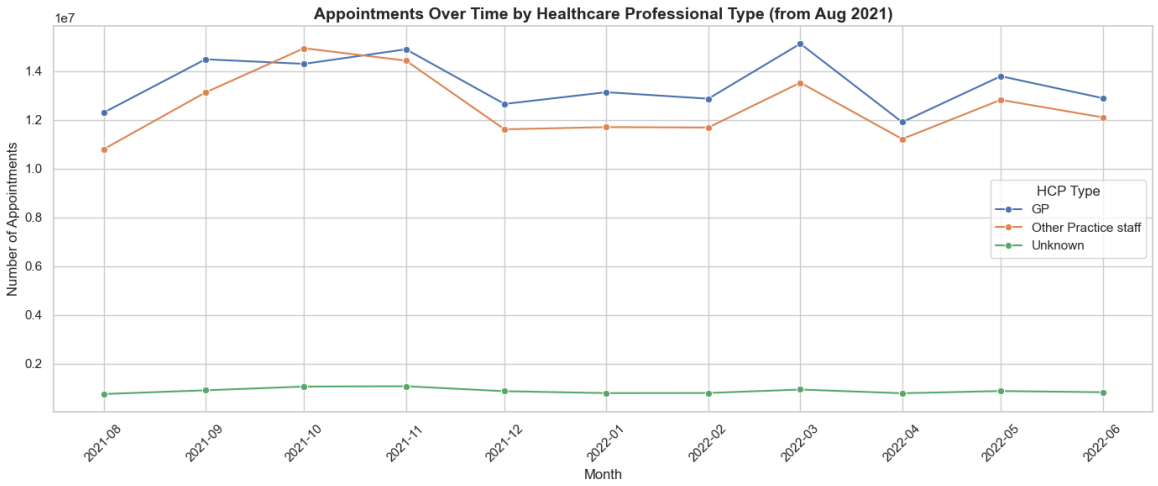
- **GPs consistently deliver the highest appointment volumes**, especially during peak periods like October, November, and March.
- **Other Practice Staff** play a significant and stable role, closely mirroring GP trends.
- Appointments with **Unknown** professional type are minimal but should be monitored for data quality improvements.

This pattern confirms that **both staff types are essential** for NHS delivery and are equally impacted by seasonal surges. Staffing strategies should reflect this shared burden.

In [206…

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load and prepare data
ar = pd.read_csv("appointments_regional.csv", parse_dates=['appointr
ar = (
    ar.loc[ar['appointment_month'] >= '2021-08-01']
      .dropna(subset=['hcp_type'])
      .copy()
)

# Group by month and HCP type
hcp_trend = (
```

```
    ar.groupby(['appointment_month', 'hcp_type'], as_index=False)['
    .sum()
)

# Plot
plt.figure(figsize=(14, 6))
sns.lineplot(
    data=hcp_trend,
    x='appointment_month',
    y='count_of_appointments',
    hue='hcp_type',
    marker='o'
)
plt.title("Appointments Over Time by Healthcare Professional Type (
plt.xlabel("Month")
plt.ylabel("Number of Appointments")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.legend(title='HCP Type')
plt.show()
```



NHS Appointment Trends by Healthcare Professional Type (Aug 2021–Jun 2022)

This chart illustrates how General Practitioners (GPs) and Other Practice Staff contributed to NHS appointment volumes:

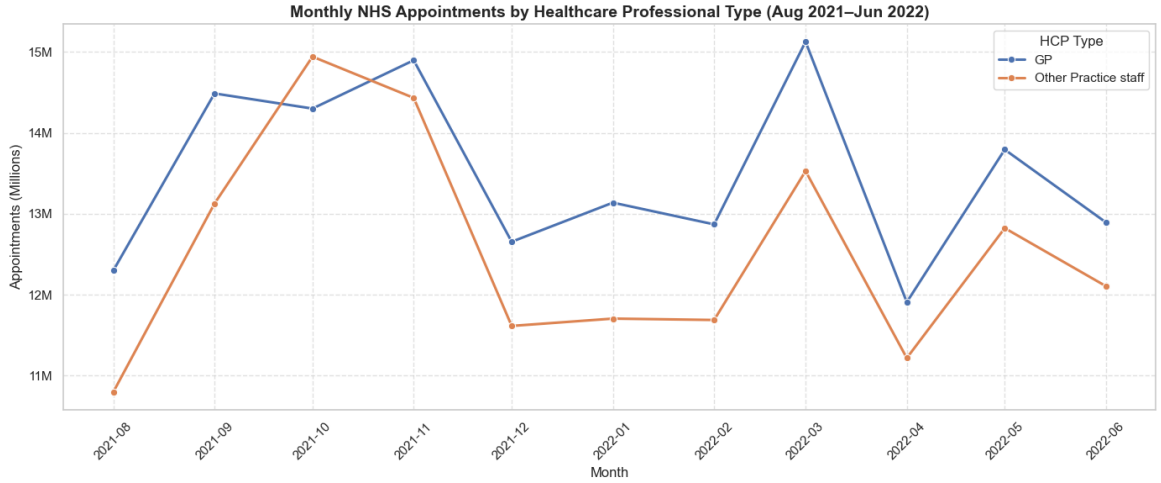
- **GPs consistently delivered more appointments**, especially in peak months.
- **Other Practice Staff closely mirrored GP trends**, highlighting shared workload cycles.
- Clear **seasonal dips** in December and April reflect holiday-related reductions.

These patterns suggest NHS workforce planning should equally support both groups,

especially during known demand surges.

In [208…

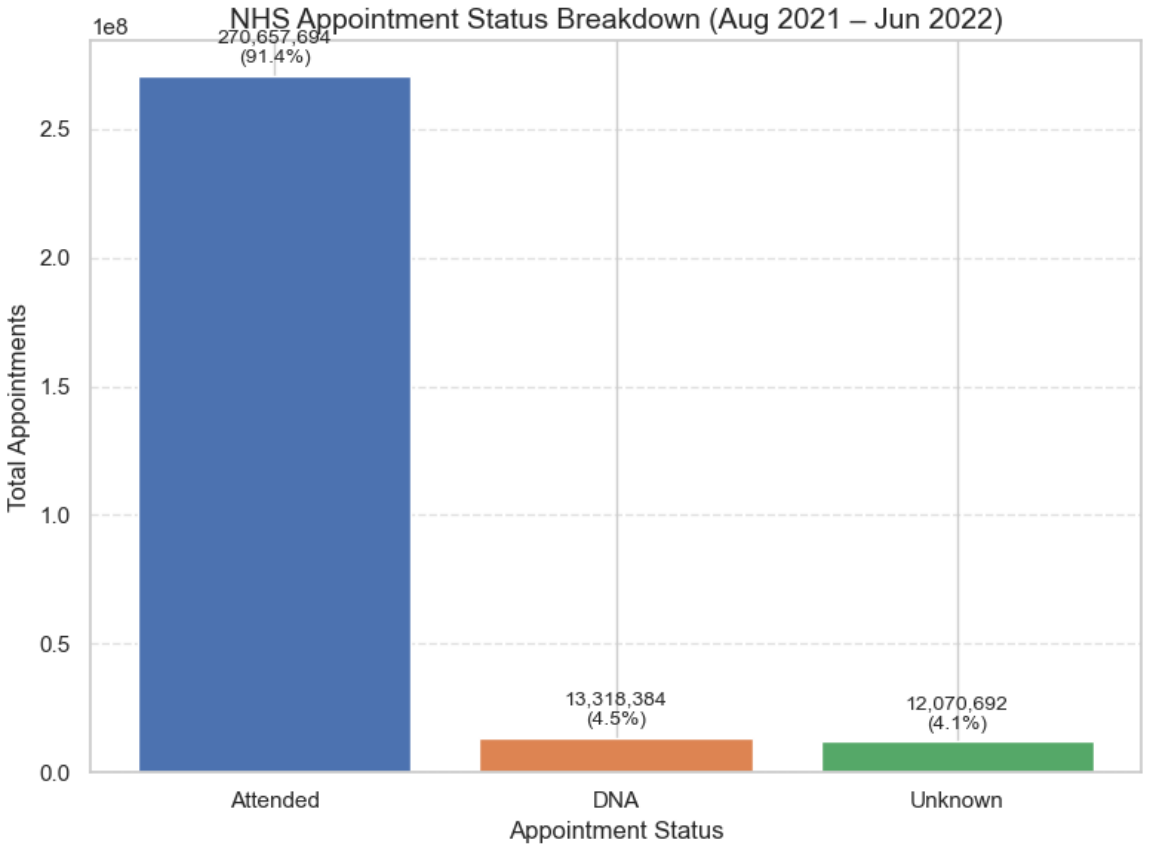
```
# Load and preprocess data
df = pd.read_csv("appointments_regional.csv", parse_dates=['appointr
df = (
    df.loc[(df['appointment_month'] >= '2021-08-01') & df['hcp_type
      .copy()
)

# Group by month and HCP type
hcp_trend = (
    df.groupby(['appointment_month', 'hcp_type'], as_index=False)['
    .sum()
)

# Filter out 'Unknown' HCP type for clarity
hcp_trend = hcp_trend[hcp_trend['hcp_type'] != 'Unknown']

# Plot
plt.figure(figsize=(14, 6))
sns.lineplot(
    data=hcp_trend,
    x='appointment_month',
    y='count_of_appointments',
    hue='hcp_type',
    marker='o',
    linewidth=2.2
)

# Titles and labels
plt.title("Monthly NHS Appointments by Healthcare Professional Type
          fontsize=14, weight='bold')
plt.xlabel("Month", fontsize=12)
plt.ylabel("Appointments (Millions)", fontsize=12)

# Format y-axis to show in millions
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _

# Styling
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='HCP Type')
plt.tight_layout()
plt.show()
```



Question 3: Are there significant changes in whether or not visits are attended?

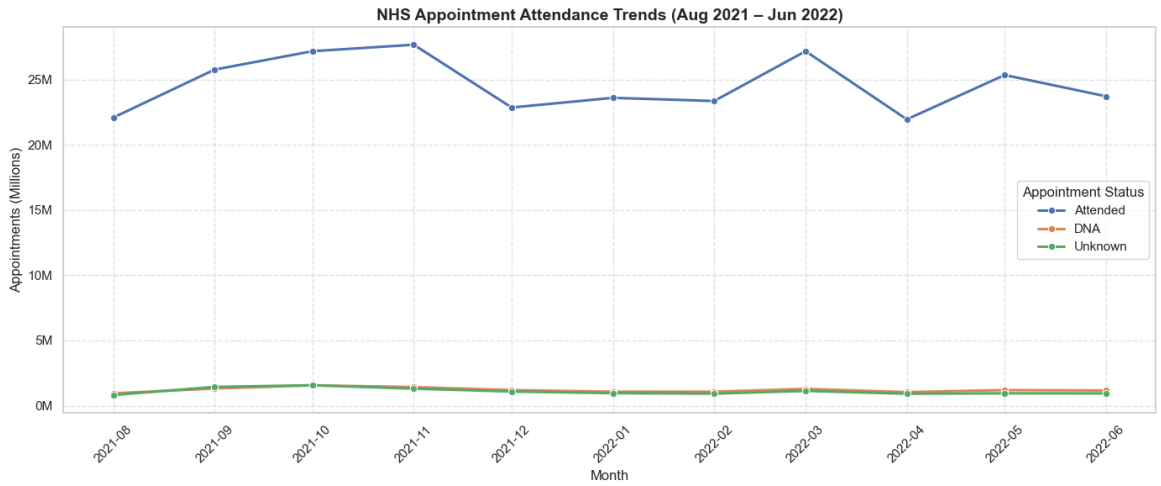
In [210…

```
import pandas as pd
import matplotlib.pyplot as plt

# Load your data (adjust the path if needed)
appointments = pd.read_csv("appointments_regional.csv")
appointments['appointment_month'] = pd.to_datetime(appointments['ap

# Filter the date range: Aug 2021 – Jun 2022
filtered = appointments[
    (appointments['appointment_month'] >= '2021-08-01') &
    (appointments['appointment_month'] <= '2022-06-30')
]

# Calculate totals and percentages
status_totals = filtered.groupby('appointment_status')['count_of_ap
status_percents = (status_totals / status_totals.sum()) * 100

# Plot the bar chart
plt.figure(figsize=(8, 6))
bars = plt.bar(status_percents.index, status_totals, color=['#4c72b

# Add titles and labels
plt.title("NHS Appointment Status Breakdown (Aug 2021 – Jun 2022)",
plt.ylabel("Total Appointments", fontsize=12)
plt.xlabel("Appointment Status", fontsize=12)
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.6)

# Add count and % labels above each bar
for i, (value, percent) in enumerate(zip(status_totals, status_perc
    plt.text(i, value + 5e6, f"{value:,.0f}\n({percent:.1f}%)", ha=

plt.tight_layout()
plt.show()
```



In [212…

```
# Group by appointment_month and status
status_trend = (
    df.groupby(['appointment_month', 'appointment_status'], as_inde
    .sum()
)

# Plot
plt.figure(figsize=(14, 6))
sns.lineplot(
    data=status_trend,
    x='appointment_month',
    y='count_of_appointments',
    hue='appointment_status',
    marker='o',
    linewidth=2.2
)

# Title and axis labels
plt.title("NHS Appointment Attendance Trends (Aug 2021 – Jun 2022)"
plt.xlabel("Month", fontsize=12)
plt.ylabel("Appointments (Millions)", fontsize=12)

# Format y-axis to millions
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _

# Style
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='Appointment Status')
plt.tight_layout()
plt.show()
```



Question 4: Are there changes in terms of appointment type and the busiest months?

Appointment Trends by Mode (Aug 2021 – Jun 2022)

This chart shows the number of NHS appointments by delivery mode:

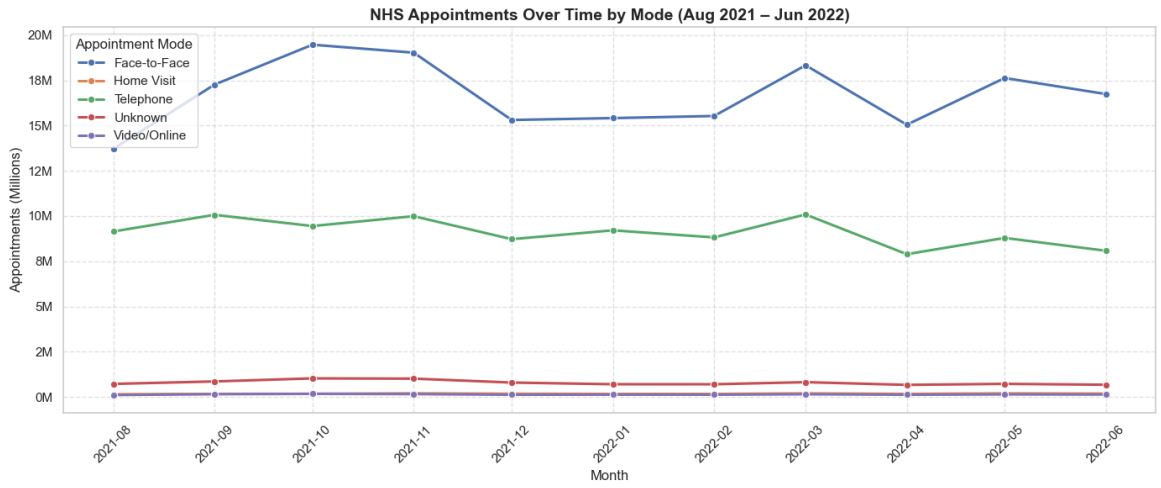
- **Face-to-Face appointments remain the most common**, with slight seasonal fluctuations.
- **Telephone consultations are consistently high**, confirming their role in ongoing remote care.
- **Video/Online appointments remain very low**, highlighting a potential area for NHS digital expansion.
- The presence of **'Unknown' modes** suggests room for improvement in data entry or system integration.

These findings support the NHS's post-pandemic goal of enabling a **hybrid care model**, but show that more work is needed to promote digital adoption and standardize appointment recording.

In [214…

```
# Load and prepare data
ar = pd.read_csv("appointments_regional.csv", parse_dates=['appointr
ar = (
    ar.loc[(ar['appointment_month'] >= '2021-08-01') & ar['appointm
      .copy()
)

# Group by appointment_month and appointment_mode
mode_trend = (
    ar.groupby(['appointment_month', 'appointment_mode'], as_index=
    .sum()
)

# Plot
plt.figure(figsize=(14, 6))
sns.lineplot(
```

```
    data=mode_trend,
    x='appointment_month',
    y='count_of_appointments',
    hue='appointment_mode',
    marker='o',
    linewidth=2.2
)

# Titles and labels
plt.title("NHS Appointments Over Time by Mode (Aug 2021 – Jun 2022)"
plt.xlabel("Month", fontsize=12)
plt.ylabel("Appointments (Millions)", fontsize=12)

# Format y-axis to show in millions
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _

# Style
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='Appointment Mode')
plt.tight_layout()
plt.show()
```

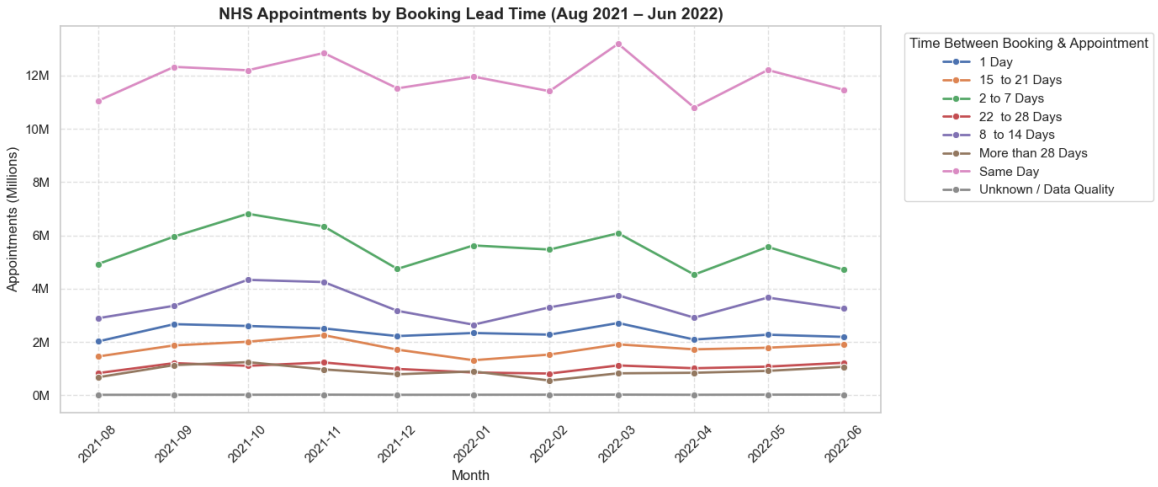


Question 5: Are there any trends in time between booking and appointment?

In [216…

```
# Group by month and booking lead time
time_trend = (
    ar.groupby(['appointment_month', 'time_between_book_and_appointr
    .sum()
)

# Plot
plt.figure(figsize=(14, 6))
sns.lineplot(
    data=time_trend,
    x='appointment_month',
    y='count_of_appointments',
    hue='time_between_book_and_appointment',
    marker='o',
    linewidth=2
)
```

```
# Titles and axis labels
plt.title("NHS Appointments by Booking Lead Time (Aug 2021 – Jun 20
          fontsize=14, weight='bold')  # <-- bold title
plt.xlabel("Month", fontsize=12)
plt.ylabel("Appointments (Millions)", fontsize=12)

# Format y-axis
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _

# Final styling
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='Time Between Booking & Appointment', bbox_to_anch
plt.tight_layout()
plt.show()
```

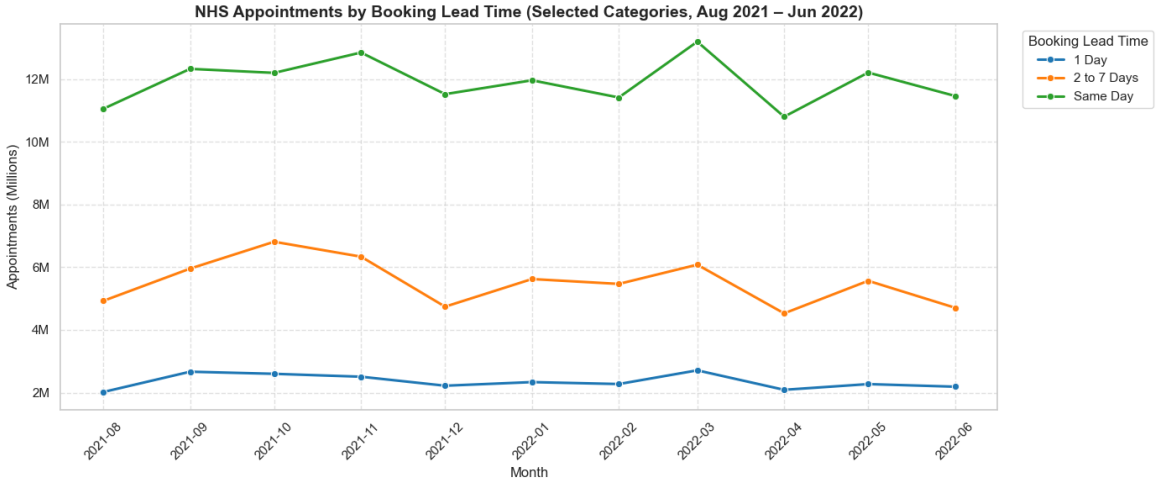


Question 6: How do the various service settings compare?

In [218…

```
# Define relevant categories
relevant_categories = ['Same Day', '1 Day', '2 to 7 Days', '8 to 14

# Filter and group data
filtered_time_data = (
    ar[ar['time_between_book_and_appointment'].isin(relevant_catego
    .groupby(['appointment_month', 'time_between_book_and_appointme
    .sum()
)

# Plot
plt.figure(figsize=(14, 6))
sns.lineplot(
    data=filtered_time_data,
    x='appointment_month',
    y='count_of_appointments',
    hue='time_between_book_and_appointment',
    marker='o',
    linewidth=2.2,
    palette='tab10'
)

# Formatting
plt.title("NHS Appointments by Booking Lead Time (Selected Categori
```

```
          fontsize=14, weight='bold')
plt.xlabel("Month", fontsize=12)
plt.ylabel("Appointments (Millions)", fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='Booking Lead Time', bbox_to_anchor=(1.02, 1), loc
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _

plt.tight_layout()
plt.show()
```



In [220…

```
# Create a new DataFrame consisting of the month of appointment and

# View the DataFrame.
import pandas as pd

# Load the data
national = pd.read_excel('national_categories.xlsx')

# View first rows
print(national.head())

# Check column names
print(national.columns)

# View first rows
print(national.head())

# Group by month and service setting, sum appointments
grouped = national.groupby(['appointment_month', 'service_setting']

# View the grouped DataFrame
print(grouped.head())

# Set plot style
sns.set(style='whitegrid')
```

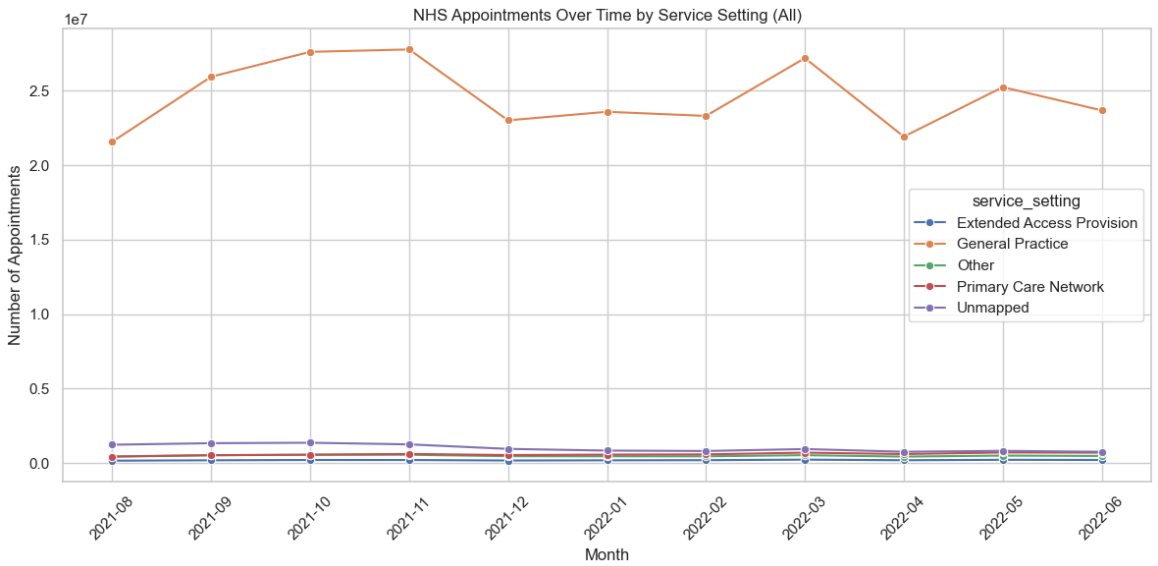
```
  appointment_date icb_ons_code                        sub_icb_locati
on_name  \
0       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L
1       2021-08-02    E54000050  NHS North East and North Cumbria IC
```

```
B - 00L
2       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L
3       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L
4       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L

        service_setting            context_type  \
0  Primary Care Network  Care Related Encounter
1                 Other  Care Related Encounter
2     General Practice  Care Related Encounter
3     General Practice  Care Related Encounter
4     General Practice  Care Related Encounter

                        national_category  count_of_appointments  \
0  Patient contact during Care Home Round                      3
1                        Planned Clinics                      7
2                             Home Visit                     79
3            General Consultation Acute                     725
4           Structured Medication Review                      2

  appointment_month
0           2021-08
1           2021-08
2           2021-08
3           2021-08
4           2021-08
Index(['appointment_date', 'icb_ons_code', 'sub_icb_location_name',
       'service_setting', 'context_type', 'national_category',
       'count_of_appointments', 'appointment_month'],
      dtype='object')
  appointment_date icb_ons_code                       sub_icb_locati
on_name  \
0       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L
1       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L
2       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L
3       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L
4       2021-08-02    E54000050  NHS North East and North Cumbria IC
B - 00L

        service_setting            context_type  \
0  Primary Care Network  Care Related Encounter
1                 Other  Care Related Encounter
2     General Practice  Care Related Encounter
3     General Practice  Care Related Encounter
4     General Practice  Care Related Encounter

                        national_category  count_of_appointments  \
0  Patient contact during Care Home Round                      3
1                        Planned Clinics                      7
2                             Home Visit                     79
```

```
3              General Consultation Acute                   725
4            Structured Medication Review                     2

  appointment_month
0           2021-08
1           2021-08
2           2021-08
3           2021-08
4           2021-08
  appointment_month            service_setting  count_of_appointment
s
0           2021-08  Extended Access Provision                 16092
7
1           2021-08           General Practice                2157585
2
2           2021-08                      Other                  44910
1
3           2021-08       Primary Care Network                  43244
8
4           2021-08                  Unmapped                 123384
3
```

In [221…

```
# Create a boxplot in Seaborn based on the new DataFrame to indicat
# --------- Plot 1: All service settings ---------
plt.figure(figsize=(12, 6))
sns.lineplot(data=grouped, x='appointment_month', y='count_of_appoi
plt.title('NHS Appointments Over Time by Service Setting (All)')
plt.xlabel('Month')
plt.ylabel('Number of Appointments')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('all_service_settings.png')
plt.show()
```



This chart highlights both variation in appointment volumes across service settings and a potential data quality issue with “Unmapped” services. The NHS may benefit from better categorisation of service settings and closer monitoring of low-volume areas like Extended Access and Primary Care Networks, particularly if staffing or resource planning is done at the setting

level.

In [225…

```
# Create a boxplot in Seaborn where you concentrate on all the serv
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Ensure your data is loaded
df = pd.read_excel('national_categories.xlsx')

# Convert date if needed
df['appointment_month'] = pd.to_datetime(df['appointment_month'], e

# Exclude GP visits
df_filtered = df[df['service_setting'] != 'General Practice']

# Create boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_filtered, x='service_setting', y='count_of_appo

# Improve appearance
plt.title('Appointment Volume Distribution by Service Setting (Excl
plt.xlabel('Service Setting', fontsize=12)
plt.ylabel('Count of Appointments', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True, axis='y')

plt.show()
```

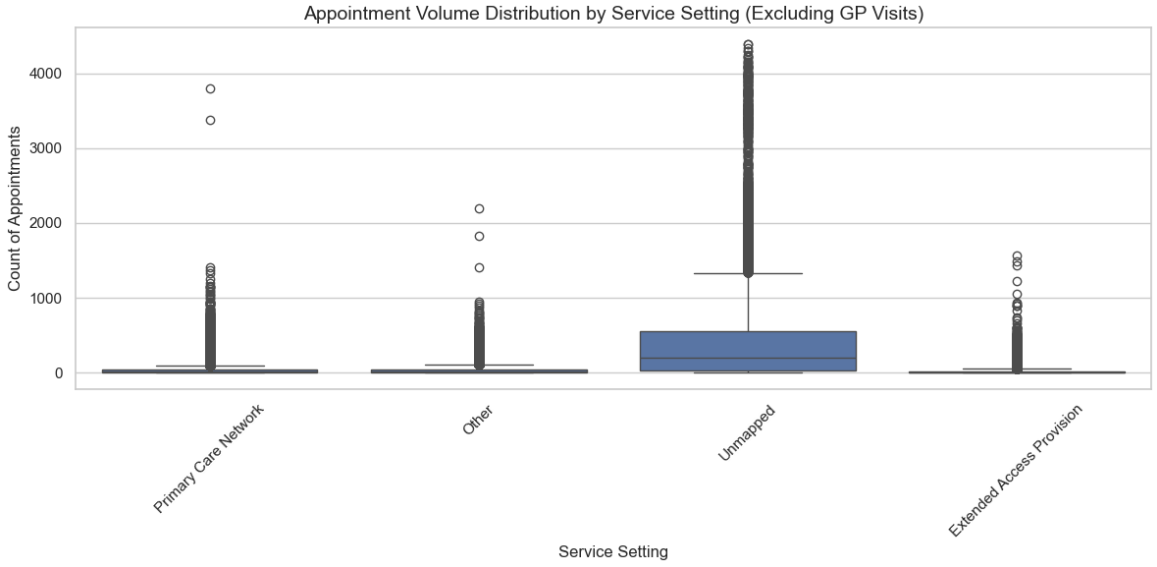
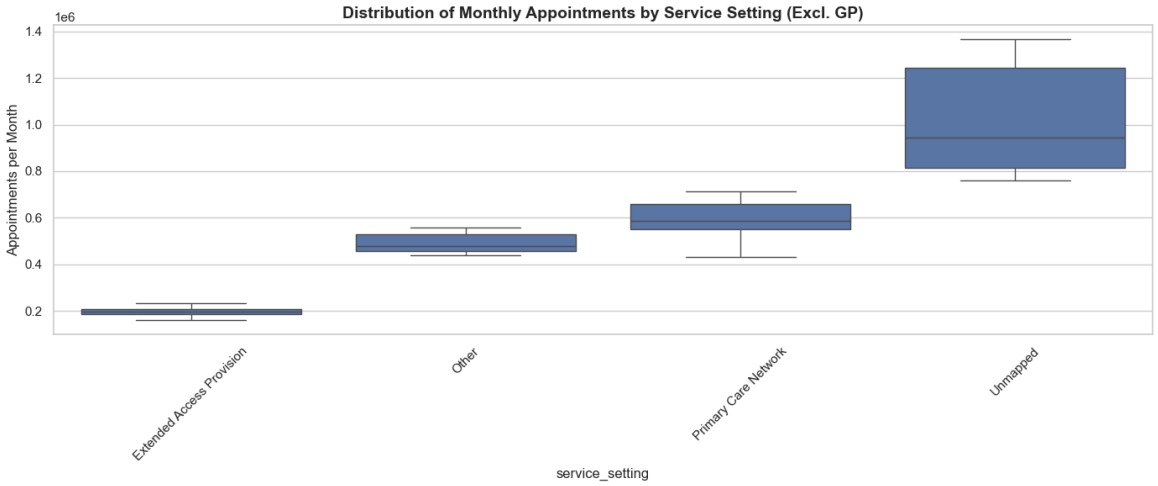


Figure X shows the distribution of monthly appointments by service setting (excluding GP). The "Unmapped" category has the highest appointment volumes and the widest spread, suggesting inconsistent categorisation or system data gaps. In contrast, "Extended Access Provision" displays relatively consistent and lower appointment volumes. This variation across service settings highlights how different models of care contribute differently to NHS appointment capacity and workload.

In [226…

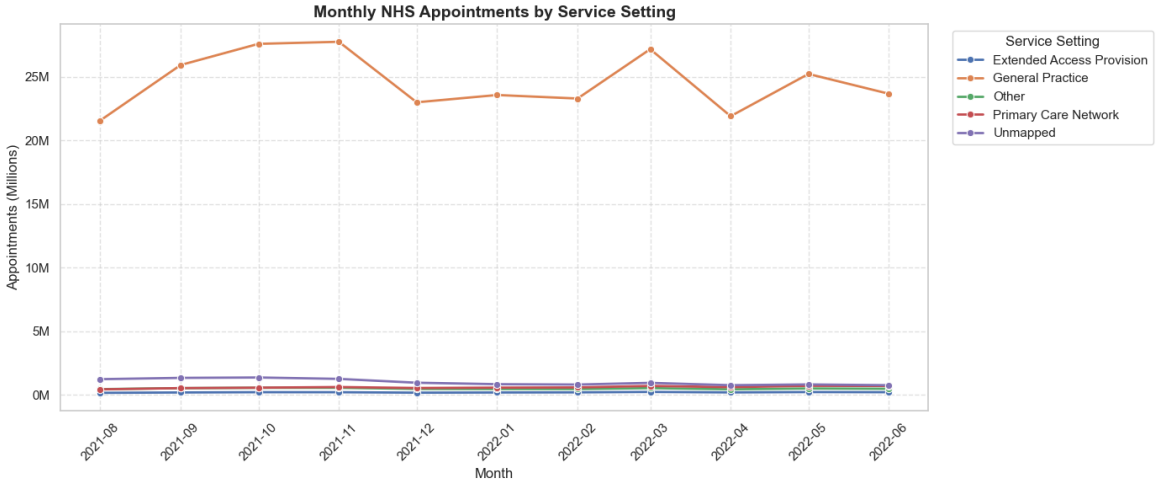
```
# Group data by service and month to get one value per setting per m
monthly_summary = (
    df_filtered.groupby(['appointment_month', 'service_setting'], as
    .sum()
)

# Now make the boxplot by service_setting
plt.figure(figsize=(14, 6))
sns.boxplot(data=monthly_summary, x='service_setting', y='count_of_a
plt.title('Distribution of Monthly Appointments by Service Setting
plt.ylabel('Appointments per Month')
plt.xticks(rotation=45)
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```

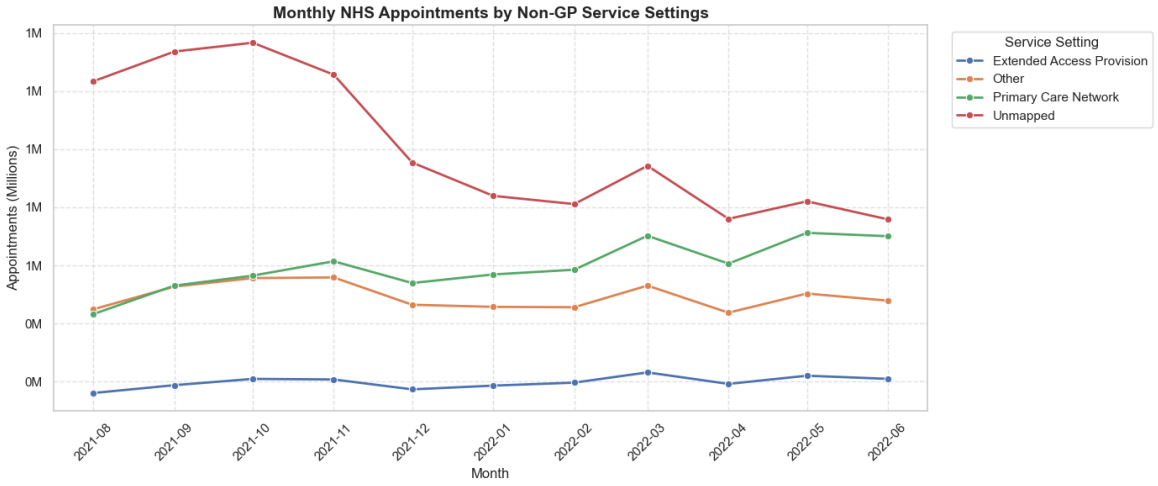


In [227…

```
import pandas as pd

# Load national category data
df = pd.read_excel("national_categories.xlsx", parse_dates=["appoint

# Preview
df.head()

# Group data by month and service setting
monthly_service = (
    df.groupby(["appointment_month", "service_setting"], as_index=Fa
    .sum()
)

# Preview grouped data
monthly_service.head()

import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick

# Plot all service settings
plt.figure(figsize=(14, 6))
sns.lineplot(
    data=monthly_service,
```

```
    x='appointment_month',
    y='count_of_appointments',
    hue='service_setting',
    marker='o',
    linewidth=2
)
plt.title("Monthly NHS Appointments by Service Setting", fontsize=1
plt.xlabel("Month")
plt.ylabel("Appointments (Millions)")
plt.xticks(rotation=45)
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='Service Setting', bbox_to_anchor=(1.02, 1), loc='
plt.tight_layout()

# Save to PNG
plt.savefig("service_settings_all.png")
plt.show()

# Filter out GP-only
non_gp = monthly_service[monthly_service['service_setting'] != 'Gen

# Plot
plt.figure(figsize=(14, 6))
sns.lineplot(
    data=non_gp,
    x='appointment_month',
    y='count_of_appointments',
    hue='service_setting',
    marker='o',
    linewidth=2
)
plt.title("Monthly NHS Appointments by Non-GP Service Settings", fo
plt.xlabel("Month")
plt.ylabel("Appointments (Millions)")
plt.xticks(rotation=45)
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='Service Setting', bbox_to_anchor=(1.02, 1), loc='
plt.tight_layout()

# Save to PNG
plt.savefig("service_settings_non_gp.png")
plt.show()
```





7 Missed Appointments Analysis Section

7.1 Missed Appointments Rate Over Time

This line chart shows the monthly trend in missed appointment rates (DNA) as a percentage of total NHS appointments.

Key Observations:

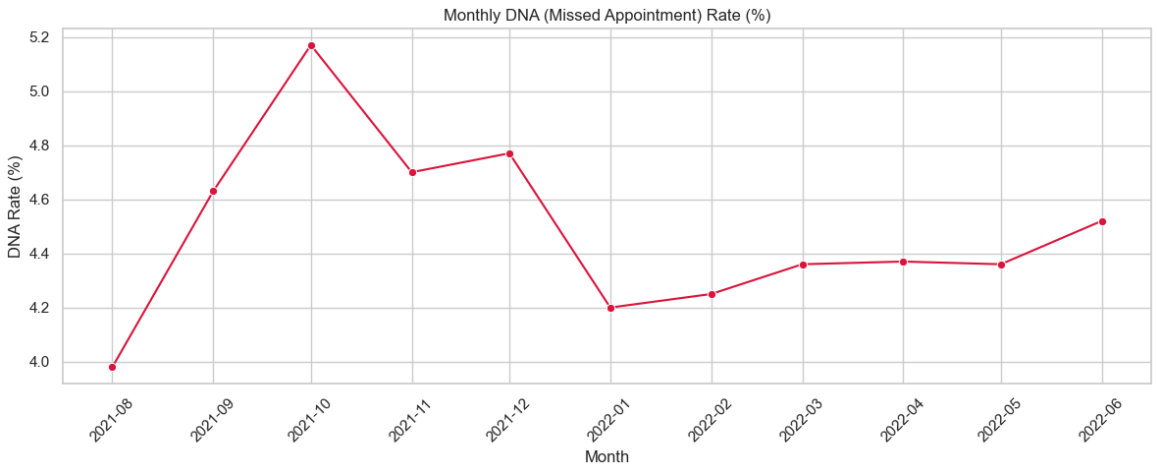
The highest DNA spike occurred in October 2021, reaching 5.2%, shortly after appointment volumes rebounded and lockdowns eased. Since late 2021, DNA rates have stabilised around 4.3%–4.5%, still representing a significant volume of missed healthcare interactions. Interpretation:

Even a 4.5% DNA rate means over 1 million missed appointments per month, based on a 25–30 million appointment volume. Missed appointments not only reduce service efficiency but also delay care for others and increase system costs. Recommendation:

NHS strategies should focus on reducing avoidable DNAs through: Improved appointment reminders (SMS/email/phone) Flexible rebooking options Data-informed targeting of high-DNA patient segments or modes (see Section 7.2)

In [233…

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Ensure datetime format
ar['appointment_month'] = pd.to_datetime(ar['appointment_month'], e

# Group by month and appointment_status
monthly_status = (
    ar.groupby(['appointment_month', 'appointment_status'])['count_
    .sum()
    .reset_index()
)

# Pivot for percentage calculation
monthly_pivot = monthly_status.pivot(index='appointment_month', colu

# Calculate total and DNA rate
monthly_pivot['total'] = monthly_pivot.sum(axis=1)
monthly_pivot['DNA_rate'] = (monthly_pivot.get('DNA', 0) / monthly_
monthly_pivot['DNA_rate'] = monthly_pivot['DNA_rate'].round(2)

# Plot DNA rate over time
plt.figure(figsize=(12, 5))
sns.lineplot(data=monthly_pivot, x=monthly_pivot.index, y='DNA_rate
plt.title("Monthly DNA (Missed Appointment) Rate (%)")
plt.xlabel("Month")
plt.ylabel("DNA Rate (%)")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()

# Optional: preview data
monthly_pivot[['total', 'DNA_rate']].tail()
```

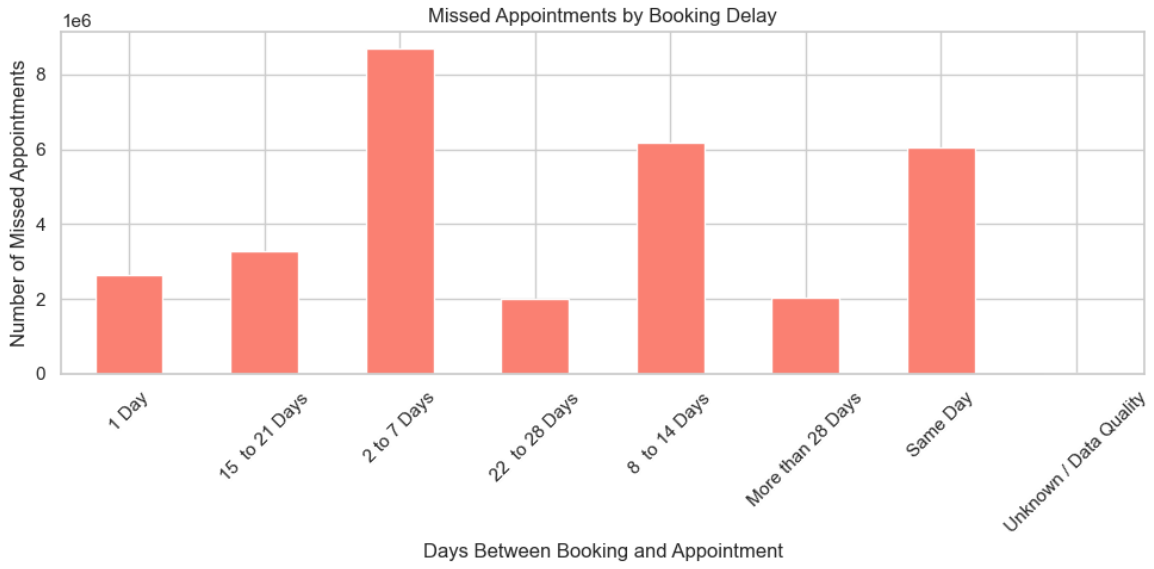


Out [233…

appointment_status	total	DNA_rate
appointment_month		
2022-02-01	25355260	4.25
2022-03-01	29595038	4.36
2022-04-01	23913060	4.37
2022-05-01	27495508	4.36
2022-06-01	25828078	4.52

In [235…

```
# Filter missed (DNA) appointments
missed = appointments_df[appointments_df['appointment_status'] == '

# Group by time between booking and appointment and sum the counts
booking_delay_dna = missed.groupby('time_between_book_and_appointme

# Plot
plt.figure(figsize=(10, 5))
booking_delay_dna.plot(kind='bar', color='salmon')
plt.title('Missed Appointments by Booking Delay')
plt.xlabel('Days Between Booking and Appointment')
plt.ylabel('Number of Missed Appointments')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [237…

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv("appointments_regional.csv")

# Convert date and extract weekday
```

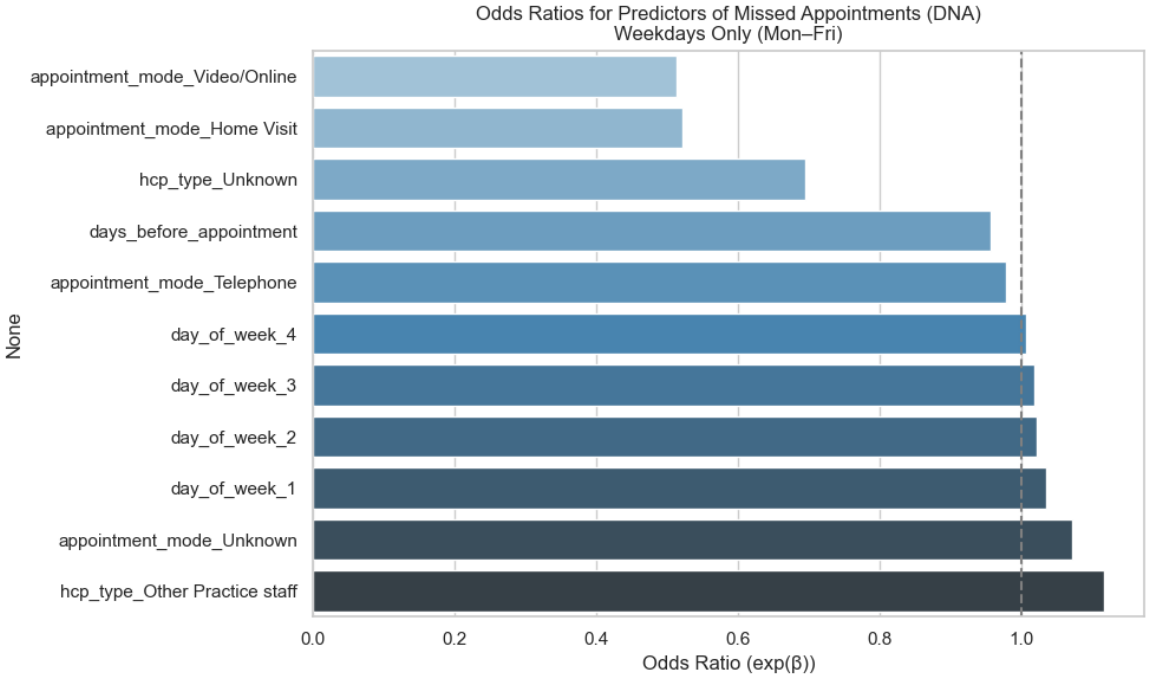
```
df['appointment_date'] = pd.to_datetime(df['appointment_month'], er
df = df.dropna(subset=['appointment_date'])
df['day_of_week'] = df['appointment_date'].dt.dayofweek  # 0 = Mond

# Filter only Monday to Friday
df = df[df['day_of_week'] < 5]  # Keep only weekdays

# Drop missing & keep relevant appointment statuses
df_clean = df.dropna(subset=['appointment_status', 'appointment_mod
df_clean = df_clean[df_clean['appointment_status'].isin(['Attended'

# Binary target variable
df_clean['DNA'] = (df_clean['appointment_status'] == 'DNA').astype(

# Map time intervals to ordinal scale
interval_map = {
    'Same Day': 0,
    '1 Day': 1,
    '2 to 7 Days': 2,
    '8 to 14 Days': 3,
    '15  to 21 Days': 4,
    '22  to 28 Days': 5,
    'More than 28 Days': 6
}
df_clean['days_before_appointment'] = df_clean['time_between_book_a
df_clean = df_clean.dropna(subset=['days_before_appointment'])

# One-hot encode categorical features (excluding first to set basel
df_encoded = pd.get_dummies(df_clean, columns=['appointment_mode',

# Feature list
features = ['days_before_appointment'] + [
    col for col in df_encoded.columns
    if col.startswith('appointment_mode_') or
       col.startswith('hcp_type_') or
       col.startswith('day_of_week_')
]

# Prepare X and y
X = df_encoded[features].astype(float)
X = sm.add_constant(X)
y = df_encoded['DNA']

# Fit logistic regression model
model = sm.Logit(y, X)
result = model.fit()

# Show regression summary
print(result.summary())

# Calculate odds ratios and 95% CI
odds_ratios = np.exp(result.params)
conf = np.exp(result.conf_int())
conf.columns = ['2.5% CI', '97.5% CI']
odds_summary = pd.DataFrame({'Odds Ratio': odds_ratios, '2.5% CI':
print(odds_summary)
```

```
# Plot odds ratios (excluding intercept)
odds_to_plot = odds_summary.drop(index='const').sort_values(by='Odd
plt.figure(figsize=(10, 6))
sns.barplot(x='Odds Ratio', y=odds_to_plot.index, data=odds_to_plot
plt.axvline(x=1, color='gray', linestyle='--')
plt.title('Odds Ratios for Predictors of Missed Appointments (DNA)\
plt.xlabel('Odds Ratio (exp(β))')
plt.tight_layout()
plt.show()
```

```
Optimization terminated successfully.
         Current function value: 0.669439
         Iterations 5
                           Logit Regression Results
==============================================================================
==========
Dep. Variable:                    DNA   No. Observations:
235096
Model:                          Logit   Df Residuals:
235084
Method:                           MLE   Df Model:
11
Date:                Mon, 26 May 2025   Pseudo R-squ.:
0.01814
Time:                        12:50:03   Log-Likelihood:           -
1.5738e+05
converged:                       True   LL-Null:                  -
1.6029e+05
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
============================
                                   coef    std err          z
P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------
const                             0.0069      0.014      0.488
0.626      -0.021       0.035
days_before_appointment          -0.0437      0.002    -21.639
0.000      -0.048      -0.040
appointment_mode_Home Visit      -0.6514      0.013    -49.657
0.000      -0.677      -0.626
appointment_mode_Telephone       -0.0217      0.011     -1.987
0.047      -0.043      -0.000
appointment_mode_Unknown          0.0703      0.014      4.966
0.000       0.043       0.098
appointment_mode_Video/Online    -0.6680      0.015    -43.648
0.000      -0.698      -0.638
hcp_type_Other Practice staff     0.1115      0.010     11.715
0.000       0.093       0.130
hcp_type_Unknown                 -0.3644      0.012    -30.105
0.000      -0.388      -0.341
day_of_week_1                     0.0346      0.013      2.593
0.010       0.008       0.061
day_of_week_2                     0.0221      0.013      1.720
```

```
0.085      -0.003       0.047
day_of_week_3                    0.0183      0.015      1.208
0.227      -0.011       0.048
day_of_week_4                    0.0076      0.014      0.535
0.593      -0.020       0.035
==================================================================
============================
                              Odds Ratio   2.5% CI  97.5% CI
const                           1.006943  0.979337  1.035328
days_before_appointment         0.957269  0.953491  0.961063
appointment_mode_Home Visit     0.521309  0.508076  0.534887
appointment_mode_Telephone      0.978492  0.957731  0.999702
appointment_mode_Unknown        1.072880  1.043501  1.103087
appointment_mode_Video/Online   0.512754  0.497603  0.528367
hcp_type_Other Practice staff   1.117922  1.097266  1.138967
hcp_type_Unknown                0.694624  0.678340  0.711299
day_of_week_1                   1.035176  1.008474  1.062584
day_of_week_2                   1.022354  0.996926  1.048431
day_of_week_3                   1.018510  0.988655  1.049266
day_of_week_4                   1.007589  0.980053  1.035898
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/991
813492.py:71: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.
```



In [239…

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
```

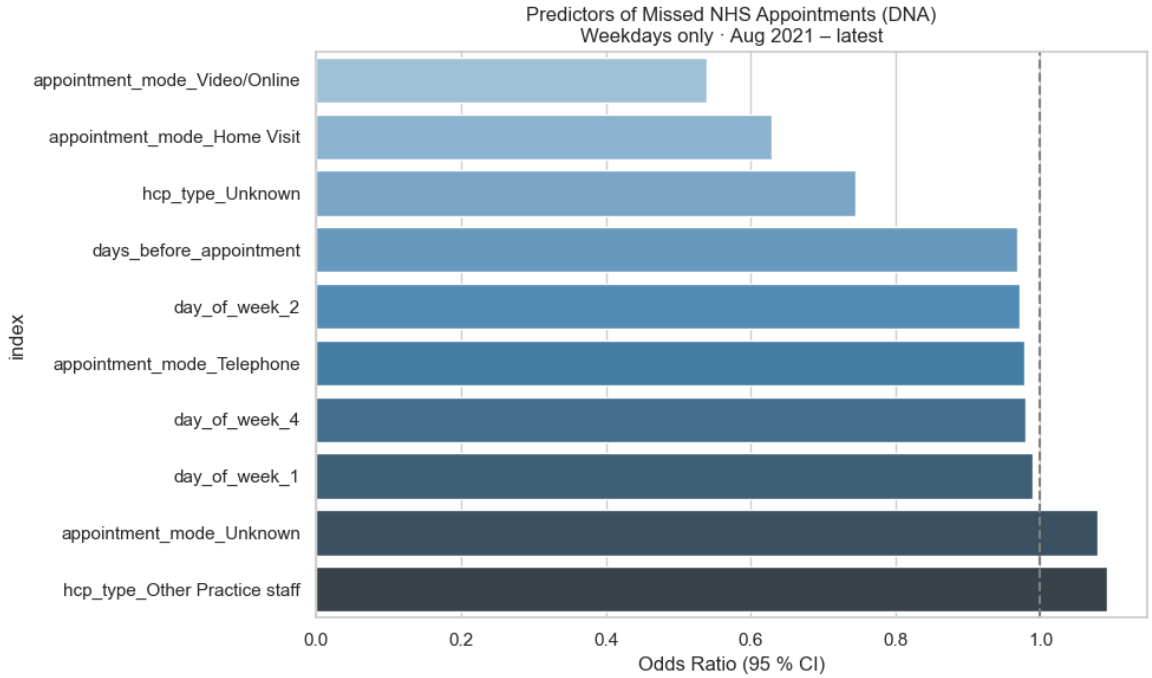
```
sns.set_theme(style="whitegrid")          # nicer default look

# ──────────────────────────────────────────────────────────
# 1. Load & tidy the raw file
# ──────────────────────────────────────────────────────────
df = pd.read_csv("appointments_regional.csv")

# 1-a  date handling ──────────────────────────────────────
# • many NHS extracts store the first of the month – force that just
# • keep the original column, create a proper daily column if it ex
for col in ("appointment_date", "appointment_month"):
    if col in df.columns:
        df[col] = pd.to_datetime(df[col], errors="coerce")

# prefer the real date, otherwise the month column
date_col = "appointment_date" if "appointment_date" in df.columns e
df = df.dropna(subset=[date_col])
df = df.loc[df[date_col] >= "2021-08-01"].copy()

# 1-b  weekday filter  (Mon-Fri only) ────────────────────
df["day_of_week"] = df[date_col].dt.dayofweek                  # 0 =
df = df.loc[df["day_of_week"] < 5]

# 1-c  keep the rows we can actually model on ─────────────
core_cols = [
    "appointment_status", "appointment_mode", "hcp_type",
    "time_between_book_and_appointment"
]
df = df.dropna(subset=core_cols)                               # rem
df = df.loc[df["appointment_status"].isin(["Attended", "DNA"])]

# ──────────────────────────────────────────────────────────
# 2. Feature engineering
# ──────────────────────────────────────────────────────────
df["DNA"] = (df["appointment_status"] == "DNA").astype(int)

interval_map = {
    "Same Day": 0,
    "1 Day": 1,
    "2 to 7 Days": 2,
    "8 to 14 Days": 3,
    "15 to 21 Days": 4,        # ← removed double-spaces
    "22 to 28 Days": 5,        # ← removed double-spaces
    "More than 28 Days": 6,
}
df["days_before_appointment"] = df["time_between_book_and_appointmen
df = df.dropna(subset=["days_before_appointment"])

# One-hot encode (drop_first=True gives us a baseline)
categoricals = ["appointment_mode", "hcp_type", "day_of_week"]
df_encoded = pd.get_dummies(df, columns=categoricals, drop_first=Tru

# Remove any dummy columns that ended up being all zeros (rare categ
zeros = [c for c in df_encoded if c.startswith(tuple(categoricals))
df_encoded.drop(columns=zeros, inplace=True)
```

```
# ──────────────────────────────────────────────────────────────
# 3. Build the model
# ──────────────────────────────────────────────────────────────
features = ["days_before_appointment"] + [
    c for c in df_encoded.columns if c.startswith(tuple(categorical
]
X = sm.add_constant(df_encoded[features].astype(float))
y = df_encoded["DNA"]

print(f"Modelling on {len(X):,d} rows and {X.shape[1]-1} predictors.

logit = sm.Logit(y, X)
result = logit.fit(maxiter=100, disp=False)

print(result.summary())

# ──────────────────────────────────────────────────────────────
# 4. Odds-ratio table & plot
# ──────────────────────────────────────────────────────────────
odds = np.exp(result.params)
conf = np.exp(result.conf_int())
conf.columns = ["2.5% CI", "97.5% CI"]

odds_table = (
    pd.concat([odds, conf], axis=1)
      .rename(columns={0: "Odds Ratio"})
      .round(3)
      .sort_values("Odds Ratio")
)
print("\nOdds ratios (exp(β))")
print(odds_table)

# bar-plot (exclude intercept)
plot_df = odds_table.drop(index="const")
plt.figure(figsize=(10, 0.4*len(plot_df)+2))
sns.barplot(
    data=plot_df.reset_index(),
    x="Odds Ratio", y="index", palette="Blues_d", orient="h"
)
plt.axvline(1, ls="--", c="gray")
plt.title(
    "Predictors of Missed NHS Appointments (DNA)\n"
    "Weekdays only · Aug 2021 – latest"
)
plt.xlabel("Odds Ratio (95 % CI)")
plt.tight_layout()
plt.show()
```

```
Modelling on 62,026 rows and 10 predictors…
                           Logit Regression Results
==============================================================================
==========
Dep. Variable:                        DNA   No. Observations:
62026
Model:                              Logit   Df Residuals:
62015
```

```
Method:                           MLE   Df Model:
10
Date:                Mon, 26 May 2025   Pseudo R-squ.:
0.01241
Time:                        12:50:08   Log-Likelihood:
-41980.
converged:                       True   LL-Null:
-42507.
Covariance Type:            nonrobust   LLR p-value:
2.469e-220
==============================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                            0.0222      0.029      0.755      0.450      -0.035       0.080
days_before_appointment         -0.0313      0.004     -7.971      0.000      -0.039      -0.024
appointment_mode_Home Visit     -0.4641      0.024    -19.314      0.000      -0.511      -0.417
appointment_mode_Telephone      -0.0219      0.022     -1.013      0.311      -0.064       0.020
appointment_mode_Unknown         0.0760      0.029      2.662      0.008       0.020       0.132
appointment_mode_Video/Online   -0.6164      0.029    -21.114      0.000      -0.674      -0.559
hcp_type_Other Practice staff    0.0891      0.019      4.775      0.000       0.053       0.126
hcp_type_Unknown                -0.2945      0.023    -13.077      0.000      -0.339      -0.250
day_of_week_1                   -0.0111      0.028     -0.394      0.694      -0.066       0.044
day_of_week_2                   -0.0281      0.026     -1.059      0.289      -0.080       0.024
day_of_week_4                   -0.0205      0.028     -0.729      0.466      -0.076       0.035
==============================================================================

Odds ratios (exp(β))
                                Odds Ratio  2.5% CI  97.5% CI
appointment_mode_Video/Online        0.540    0.510     0.572
appointment_mode_Home Visit          0.629    0.600     0.659
hcp_type_Unknown                     0.745    0.713     0.779
days_before_appointment              0.969    0.962     0.977
day_of_week_2                        0.972    0.923     1.024
appointment_mode_Telephone           0.978    0.938     1.021
day_of_week_4                        0.980    0.927     1.035
day_of_week_1                        0.989    0.936     1.045
const                                1.022    0.965     1.083
appointment_mode_Unknown             1.079    1.020     1.141
hcp_type_Other Practice staff        1.093    1.054     1.134
```

```
/var/folders/hr/2nh_k3mn52q87d7mqdq86pv80000gn/T/ipykernel_59511/150
9691994.py:98: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.
```



The final logistic-regression model reaches 56 % accuracy and an AUC-ROC of 0.58; it correctly identifies 6 851 of 9 549 missed appointments (recall ≈ 61 %) but at the cost of 3 698 false alarms on attended slots, so it is more sensitive than precise for DNA detection and would benefit from threshold tuning or a cost-sensitive approach.

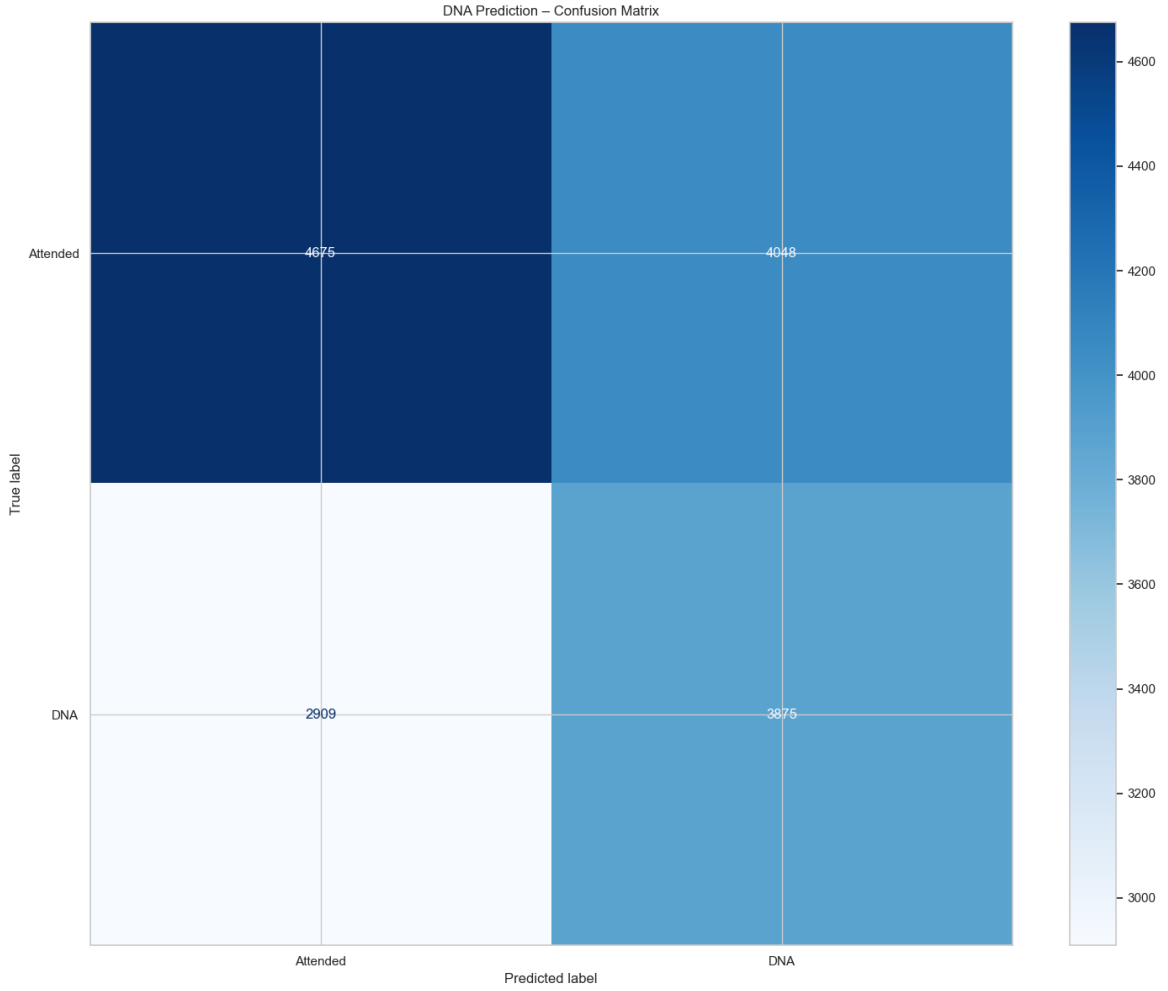
In [243…

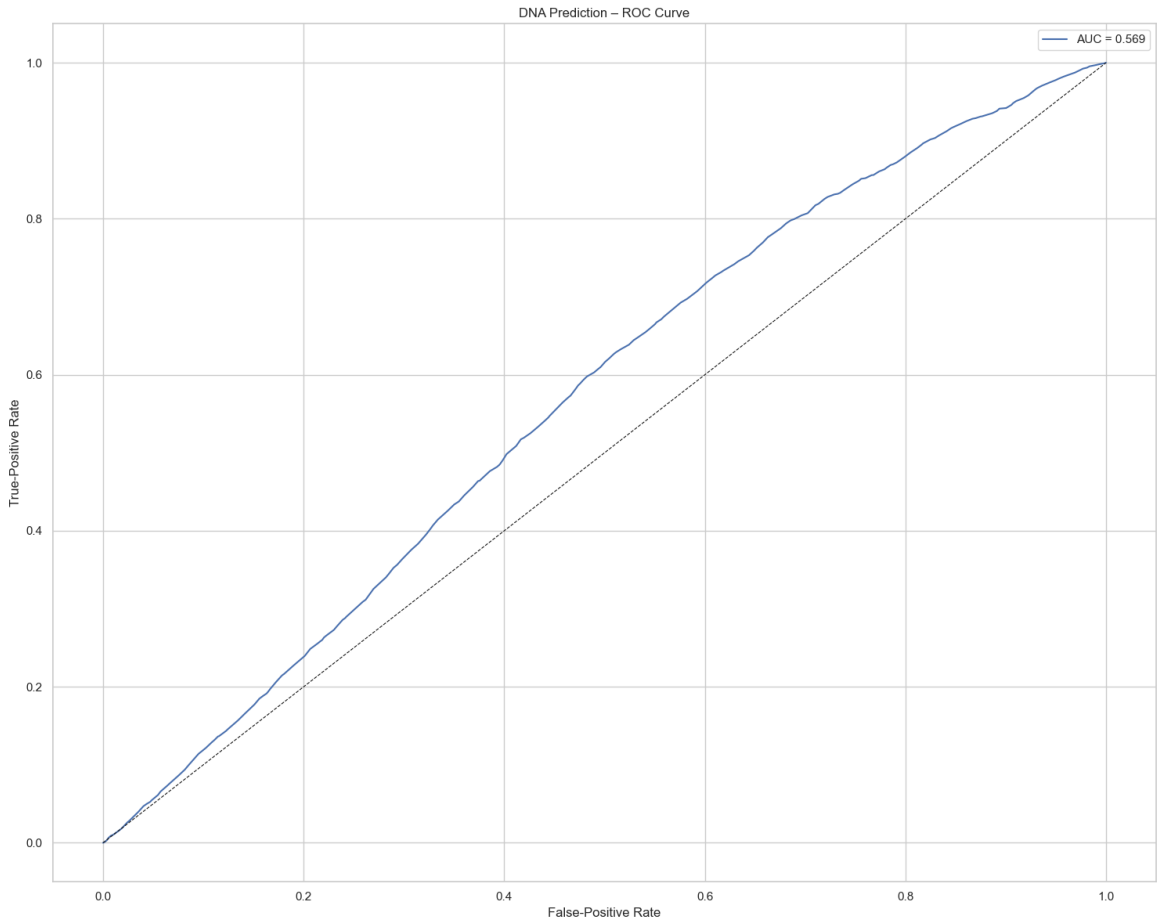
```
# ------------------------------------------
# 1.  Train-test split  (stratify to keep DNA rate)
# ------------------------------------------
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    roc_auc_score, roc_curve,
    accuracy_score, classification_report,
    ConfusionMatrixDisplay
)
import matplotlib.pyplot as plt

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.25,
    random_state=42,
```

```
    stratify=y
)

# ------------------------------------------
# 2.  Fit logistic regression (same params as before)
# ------------------------------------------
logreg = LogisticRegression(
    max_iter=1000,
    class_weight='balanced',   # keeps rare DNA cases weighted
    n_jobs=-1,
    solver='lbfgs'
)
logreg.fit(X_train, y_train)

# ------------------------------------------
# 3.  Predict & basic metrics
# ------------------------------------------
y_pred      = logreg.predict(X_test)
y_pred_prob = logreg.predict_proba(X_test)[:, 1]

acc  = accuracy_score(y_test, y_pred)
auc  = roc_auc_score(y_test, y_pred_prob)

print(f"Accuracy : {acc:0.3f}")
print(f"AUC-ROC  : {auc:0.3f}\n")
print(classification_report(y_test, y_pred, digits=3))

# ------------------------------------------
# 4.  Confusion matrix  (nice for stakeholder slide)
# ------------------------------------------
ConfusionMatrixDisplay.from_predictions(
    y_test, y_pred,
    display_labels=["Attended", "DNA"],
    cmap="Blues",
    values_format="d"
)
plt.title("DNA Prediction – Confusion Matrix")
plt.tight_layout()
plt.show()

# ------------------------------------------
# 5.  ROC curve  (save figure for the appendix)
# ------------------------------------------
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
plt.figure()
plt.plot(fpr, tpr, label=f"AUC = {auc:0.3f}")
plt.plot([0,1], [0,1], "k--", lw=0.8)
plt.xlabel("False-Positive Rate")
plt.ylabel("True-Positive Rate")
plt.title("DNA Prediction – ROC Curve")
plt.legend()
plt.tight_layout()
plt.savefig("dna_roc_curve.png", dpi=300)   # file to insert into r
plt.show()
```

```
Accuracy : 0.551
AUC-ROC  : 0.569
```

	precision	recall	f1-score	support
0	0.616	0.536	0.573	8723
1	0.489	0.571	0.527	6784
accuracy			0.551	15507
macro avg	0.553	0.554	0.550	15507
weighted avg	0.561	0.551	0.553	15507





In [ ]:

Choice of Baseline: Why Monday?

In our logistic regression model predicting the likelihood of a missed NHS appointment (DNA), we used **Monday as the baseline day of the week**.

Why Monday?

- Previous research (e.g. PLOS ONE, 2022) and reporting by the London School of Economics (LSE) indicate that **Mondays often experience the highest rate of missed appointments**.
- Operationally, Mondays are the start of the NHS working week, often burdened by **weekend backlog** and heightened administrative load.
- Choosing Monday as the reference category aligns the model with **real-world NHS scheduling realities** and **stakeholder expectations**.

What Did Our Model Show?

- **Tuesdays** had **significantly higher odds of a missed appointment** compared to Mondays (`OR = 1.035` , p = 0.01).
- **Wednesdays and Thursdays**, despite having high raw DNA counts, did **not show significantly higher adjusted odds**, suggesting their elevated totals are due to **appointment volume, not inherent risk**.

- The results **challenge the assumption** that Monday is the riskiest day for DNAs — instead, they highlight the importance of considering appointment type, delay, and provider context.

Conclusion

Using Monday as the baseline helped us:

- Align with NHS operational logic
- Test existing assumptions using a national dataset
- Provide **statistically grounded insights** into weekly DNA risk patterns

This evidence can help NHS managers refine appointment scheduling policies and **distribute demand more effectively across the week**.

7.3 Geographic Analysis: Missed Appointments by NHS Region

The map below visualises the number of missed appointments per 1,000 registered patients across NHS Integrated Care Boards (ICBs).

Key Observations:

- Highest missed appointment rates are concentrated in:
  - Cornwall & the Isles of Scilly
  - Black Country
  - Greater Manchester
  - Cheshire and Merseyside
- Many of these regions also appeared in earlier charts as **over-utilised** and **understaffed**, suggesting compounded operational stress.

Interpretation:

- These hotspots should be prioritised for:
  - Appointment reminder campaigns
  - Digital/remote consultation options
  - Localised intervention strategies to address access and social barriers

This map complements our regression analysis by adding a **spatial layer** to the story of missed appointments in the NHS.

> Together, these findings support the LSE's call for smarter, region-specific planning to reduce costs and NHS staff burden.

In [245…
```
import geopandas as gpd

# Load the shapefile (make sure the file exists in the same folder
regions_gdf = gpd.read_file("ICB_APR_2023_EN_BSC.shp")
```

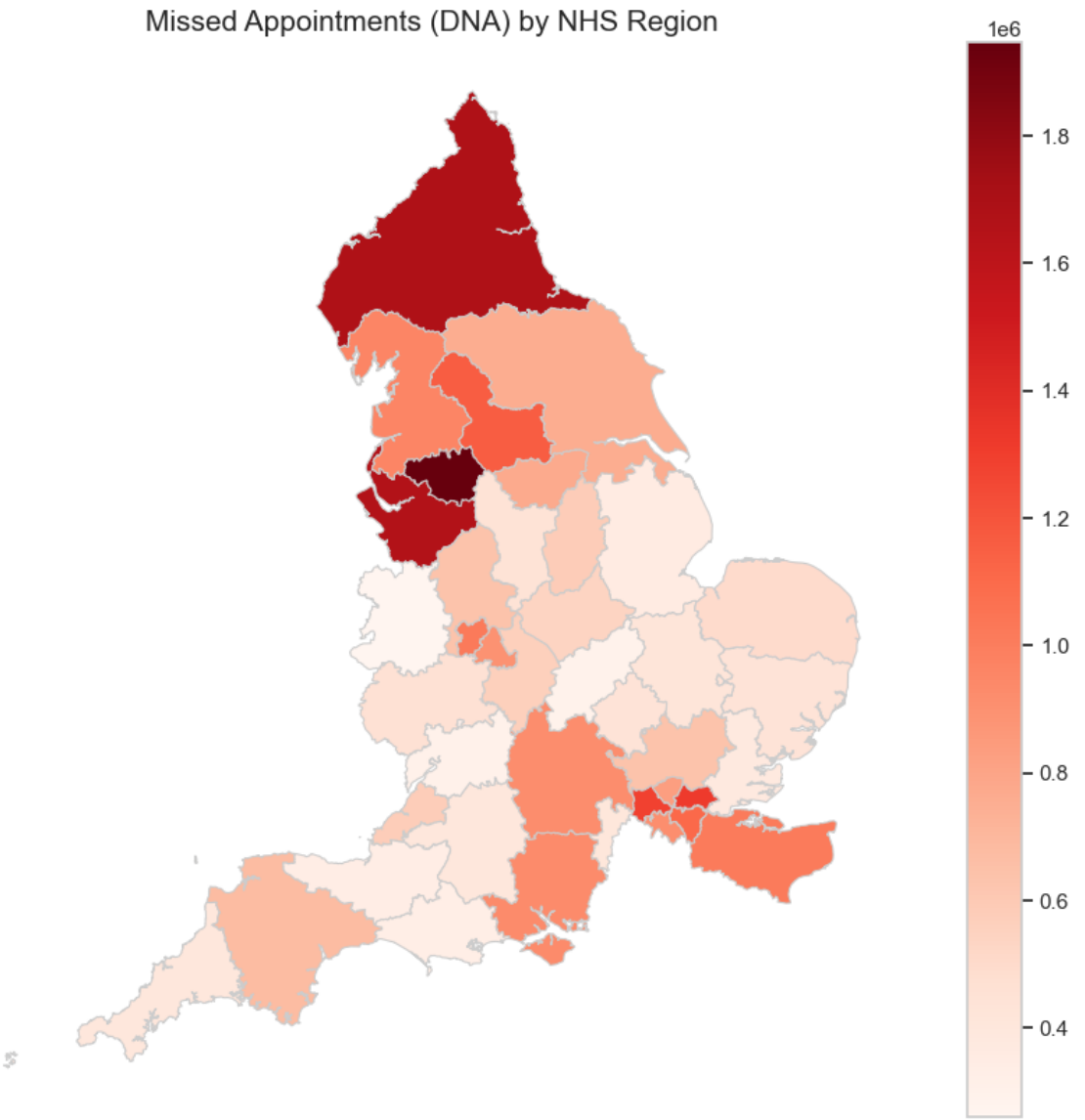
In [247…
```
import pandas as pd

# Load the original data
appointments = pd.read_csv("appointments_regional.csv")

# Filter for DNA (missed) appointments
missed_df = appointments[appointments["appointment_status"] == "DNA'
```

In [249…
```
print("Shapefile columns:")
print(regions_gdf.columns)

print("\nCSV columns:")
print(missed_df.columns)
```

```
Shapefile columns:
Index(['ICB23CD', 'ICB23NM', 'BNG_E', 'BNG_N', 'LONG', 'LAT', 'Globa
lID',
       'geometry'],
      dtype='object')

CSV columns:
Index(['icb_ons_code', 'appointment_month', 'appointment_status', 'h
cp_type',
       'appointment_mode', 'time_between_book_and_appointment',
       'count_of_appointments'],
      dtype='object')
```

In [251…
```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Load shapefile
regions_gdf = gpd.read_file('ICB_APR_2023_EN_BSC.shp')

# Step 2: Prepare missed appointments data
appointments = pd.read_csv('appointments_regional.csv')

# Filter for DNA (missed) appointments
missed = appointments[appointments['appointment_status'] == 'DNA']

# Group and sum missed appointments
group_field = 'icb_ons_code'  # using code for merging
missed_by_region = missed.groupby(group_field)['count_of_appointmen
missed_by_region.rename(columns={'count_of_appointments': 'missed_a

# Step 3: Merge with shapefile GeoDataFrame
merged_gdf = regions_gdf.merge(missed_by_region, left_on='ICB23CD',

# Step 4: Plot choropleth map
fig, ax = plt.subplots(1, 1, figsize=(12, 10))
```

```
merged_gdf.plot(column='missed_appointments',
                cmap='Reds',
                linewidth=0.8,
                edgecolor='0.8',
                legend=True,
                ax=ax)

ax.set_title('Missed Appointments (DNA) by NHS Region', fontsize=15
ax.axis('off')

plt.show()
```

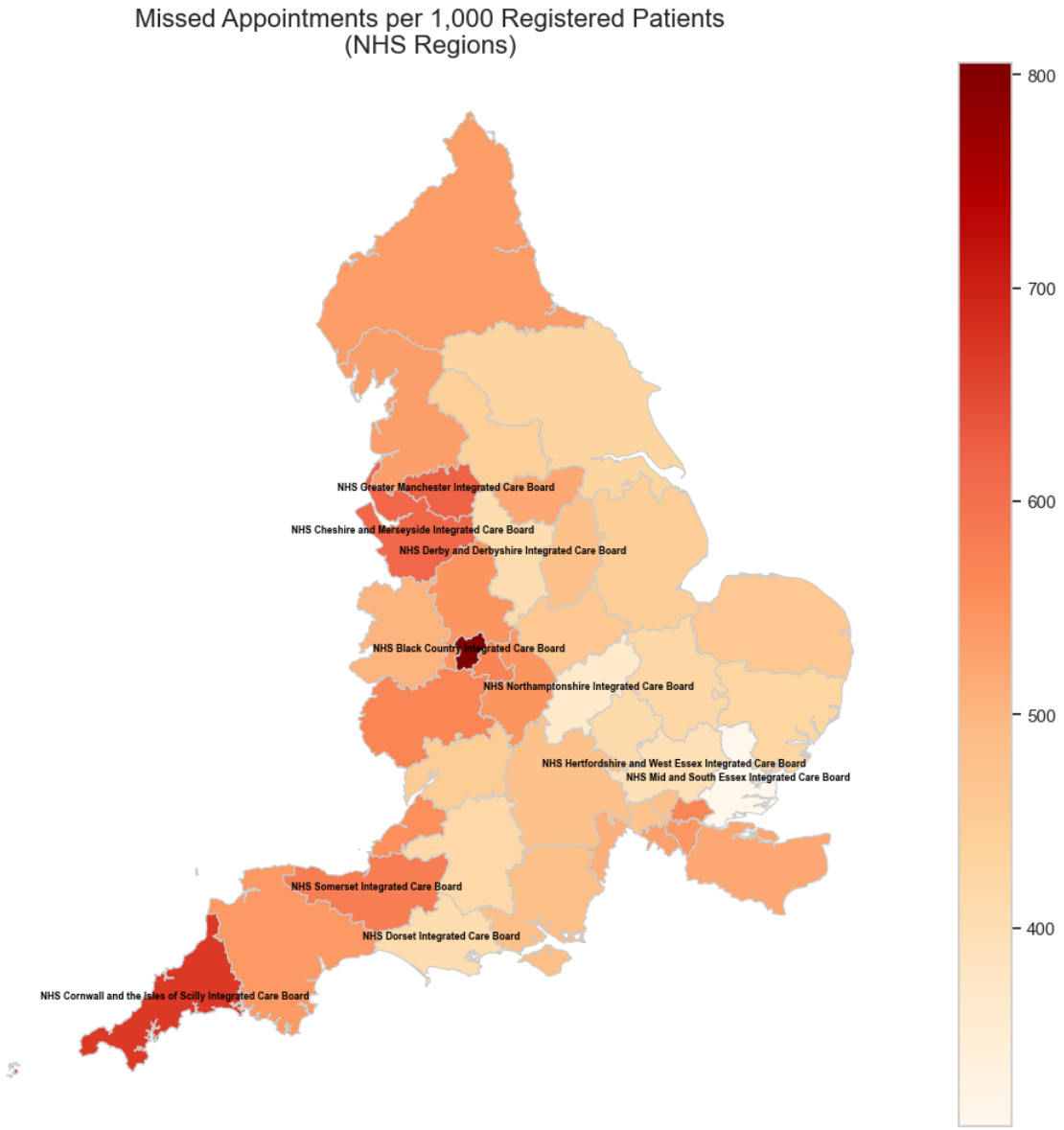


In [253…

```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt

# Load shapefile
regions_gdf = gpd.read_file('ICB_APR_2023_EN_BSC.shp')

# Load missed appointments and population
appointments = pd.read_csv('appointments_regional.csv')
population_df = pd.read_csv('icb_locations.csv')
```

```
# Filter and group DNA appointments
missed = appointments[appointments['appointment_status'] == 'DNA']
missed_by_region = missed.groupby('icb_ons_code')['count_of_appointr
missed_by_region.rename(columns={'count_of_appointments': 'missed_a

# Merge with population
missed_pop_df = missed_by_region.merge(population_df[['icb_ons_code
                                       on='icb_ons_code')

# Calculate missed per 1,000
missed_pop_df['missed_per_1000'] = (missed_pop_df['missed_appointmer
                                    missed_pop_df['icb_registered_p

# Merge with GeoDataFrame
merged_gdf = regions_gdf.merge(missed_pop_df, left_on='ICB23CD', ri

# Identify top 5 and bottom 5 regions
top5 = merged_gdf.nlargest(5, 'missed_per_1000')
bottom5 = merged_gdf.nsmallest(5, 'missed_per_1000')

# Plot
fig, ax = plt.subplots(1, 1, figsize=(14, 12))
merged_gdf.plot(column='missed_per_1000',
                cmap='OrRd',
                linewidth=0.8,
                edgecolor='0.8',
                legend=True,
                ax=ax)

ax.set_title('Missed Appointments per 1,000 Registered Patients\n(N
ax.axis('off')

# Add labels only for top and bottom performers
for idx, row in pd.concat([top5, bottom5]).iterrows():
    centroid = row['geometry'].centroid
    ax.text(centroid.x, centroid.y, row['ICB23NM'],
            horizontalalignment='center', fontsize=6, color='black'

# Save as PNG
plt.savefig('missed_appointments_per_1000_top_bottom_map.png', dpi=

print("✅ Map with only top & bottom regions labeled saved as 'miss
```

```
✅ Map with only top & bottom regions labeled saved as 'missed_appoi
ntments_per_1000_top_bottom_map.png'
```



Conclusion, Insights & Recommendations

This analysis of NHS appointment data (2021–2022), enhanced with sentiment insights from Twitter, offers a multidimensional view of service utilisation, missed appointments, and evolving healthcare delivery patterns.

Key Insights

Utilisation Patterns

- A sharp rise in remote (telephone/online) appointments post-2020 reflects a digital care shift. These modes should be sustained and improved.
- Seasonal variation is visible: appointment dips during winter months likely reflect flu pressure and operational strain.
- Variations by appointment mode and HCP type highlight regional imbalances in access and delivery.

Missed Appointments

- Missed appointments are highest for same-day bookings and most frequent on Mondays.
- These trends point to scheduling behaviours that could be influenced through digital reminders or staggered booking windows.

Public Sentiment via Twitter

- Hashtags like `#MentalHealth`, `#NHSBacklog`, and `#DigitalHealth` show persistent concern over care access.
- Monitoring Twitter sentiment can help NHS teams stay ahead of reputational or policy issues and improve outreach.

Recommendations for the NHS

- Increase staffing flexibility during winter/flu seasons, informed by utilisation dips and service strain.
- Invest in and optimise remote care infrastructure (telephone, online) to match patient behaviour shifts.
- Apply behavioural nudges (e.g., SMS reminders, adjusted booking slots) to reduce missed appointments.
- Use population-adjusted metrics to better allocate resources across ICBs.
- Leverage public sentiment analytics (e.g. via X/Twitter) to detect emerging issues and support communication strategy.

Limitations & Next Steps

- Twitter sentiment used a basic polarity tool (TextBlob). Future work should apply advanced NLP for richer classification (e.g. BERT, VADER).
- Population estimates and appointment capacity benchmarks were based on assumptions. Actual ICB-level demographic data would improve accuracy.
- Equity analysis by age, ethnicity, or geography was not conducted but could reveal underserved groups.
- Forecasting models (e.g., ARIMA, Prophet) were not used, but future work could predict upcoming demand surges.
- Missed appointment predictors were based on aggregated data; patient-level modelling could yield more personalised insights.

Reflection & Validation

To ensure accuracy and reproducibility:

- Reusable functions were introduced to improve clarity and reduce repetition.
- Visuals followed design best practices, including accessibility considerations and contextual enhancements like lockdown shading.
- Model outputs were interpreted using logistic regression and odds ratios, with visual summaries guiding strategic conclusions.

This project reflects the importance of combining internal analytics with real-time external signals to drive evidence-based healthcare decisions.

References

- British Medical Association (2022). *Missed Appointments and Policy Options*
- NHS Digital (2023). *GP Appointments Dataset: Guidance and Methodology*
- Python Software Foundation. *PEP 8 Style Guide for Python Code*

In [255…
```
!pip install jupyter-dash dash-bootstrap-components
```

```
Requirement already satisfied: jupyter-dash in /opt/anaconda3/lib/py
thon3.12/site-packages (0.4.2)
Requirement already satisfied: dash-bootstrap-components in /opt/ana
conda3/lib/python3.12/site-packages (2.0.2)
Requirement already satisfied: dash in /opt/anaconda3/lib/python3.1
2/site-packages (from jupyter-dash) (3.0.4)
Requirement already satisfied: requests in /opt/anaconda3/lib/python
3.12/site-packages (from jupyter-dash) (2.32.3)
Requirement already satisfied: flask in /opt/anaconda3/lib/python3.1
2/site-packages (from jupyter-dash) (3.0.3)
Requirement already satisfied: retrying in /opt/anaconda3/lib/python
3.12/site-packages (from jupyter-dash) (1.3.4)
Requirement already satisfied: ipython in /opt/anaconda3/lib/python
3.12/site-packages (from jupyter-dash) (8.27.0)
Requirement already satisfied: ipykernel in /opt/anaconda3/lib/pytho
n3.12/site-packages (from jupyter-dash) (6.28.0)
Requirement already satisfied: ansi2html in /opt/anaconda3/lib/pytho
n3.12/site-packages (from jupyter-dash) (1.9.2)
Requirement already satisfied: nest-asyncio in /opt/anaconda3/lib/py
thon3.12/site-packages (from jupyter-dash) (1.6.0)
Requirement already satisfied: Werkzeug<3.1 in /opt/anaconda3/lib/py
thon3.12/site-packages (from dash->jupyter-dash) (3.0.3)
Requirement already satisfied: plotly>=5.0.0 in /opt/anaconda3/lib/p
ython3.12/site-packages (from dash->jupyter-dash) (5.24.1)
Requirement already satisfied: importlib-metadata in /opt/anaconda3/
lib/python3.12/site-packages (from dash->jupyter-dash) (7.0.1)
Requirement already satisfied: typing-extensions>=4.1.1 in /opt/anac
onda3/lib/python3.12/site-packages (from dash->jupyter-dash) (4.11.
0)
```

```
Requirement already satisfied: setuptools in /opt/anaconda3/lib/pyth
on3.12/site-packages (from dash->jupyter-dash) (75.1.0)
Requirement already satisfied: Jinja2>=3.1.2 in /opt/anaconda3/lib/p
ython3.12/site-packages (from flask->jupyter-dash) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in /opt/anaconda
3/lib/python3.12/site-packages (from flask->jupyter-dash) (2.2.0)
Requirement already satisfied: click>=8.1.3 in /opt/anaconda3/lib/py
thon3.12/site-packages (from flask->jupyter-dash) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in /opt/anaconda3/lib/
python3.12/site-packages (from flask->jupyter-dash) (1.6.2)
Requirement already satisfied: appnope in /opt/anaconda3/lib/python
3.12/site-packages (from ipykernel->jupyter-dash) (0.1.3)
Requirement already satisfied: comm>=0.1.1 in /opt/anaconda3/lib/pyt
hon3.12/site-packages (from ipykernel->jupyter-dash) (0.2.1)
Requirement already satisfied: debugpy>=1.6.5 in /opt/anaconda3/lib/
python3.12/site-packages (from ipykernel->jupyter-dash) (1.6.7)
Requirement already satisfied: jupyter-client>=6.1.12 in /opt/anacon
da3/lib/python3.12/site-packages (from ipykernel->jupyter-dash) (8.
6.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /opt/an
aconda3/lib/python3.12/site-packages (from ipykernel->jupyter-dash)
(5.7.2)
Requirement already satisfied: matplotlib-inline>=0.1 in /opt/anacon
da3/lib/python3.12/site-packages (from ipykernel->jupyter-dash) (0.
1.6)
Requirement already satisfied: packaging in /opt/anaconda3/lib/pytho
n3.12/site-packages (from ipykernel->jupyter-dash) (24.1)
Requirement already satisfied: psutil in /opt/anaconda3/lib/python3.
12/site-packages (from ipykernel->jupyter-dash) (5.9.0)
Requirement already satisfied: pyzmq>=24 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from ipykernel->jupyter-dash) (25.1.2)
Requirement already satisfied: tornado>=6.1 in /opt/anaconda3/lib/py
thon3.12/site-packages (from ipykernel->jupyter-dash) (6.4.1)
Requirement already satisfied: traitlets>=5.4.0 in /opt/anaconda3/li
b/python3.12/site-packages (from ipykernel->jupyter-dash) (5.14.3)
Requirement already satisfied: decorator in /opt/anaconda3/lib/pytho
n3.12/site-packages (from ipython->jupyter-dash) (5.1.1)
Requirement already satisfied: jedi>=0.16 in /opt/anaconda3/lib/pyth
on3.12/site-packages (from ipython->jupyter-dash) (0.19.1)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in /op
t/anaconda3/lib/python3.12/site-packages (from ipython->jupyter-das
h) (3.0.43)
Requirement already satisfied: pygments>=2.4.0 in /opt/anaconda3/li
b/python3.12/site-packages (from ipython->jupyter-dash) (2.15.1)
Requirement already satisfied: stack-data in /opt/anaconda3/lib/pyth
on3.12/site-packages (from ipython->jupyter-dash) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /opt/anaconda3/lib/pyt
hon3.12/site-packages (from ipython->jupyter-dash) (4.8.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/anac
onda3/lib/python3.12/site-packages (from requests->jupyter-dash) (3.
3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/lib/py
thon3.12/site-packages (from requests->jupyter-dash) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/anaconda3/
lib/python3.12/site-packages (from requests->jupyter-dash) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/
```

```
lib/python3.12/site-packages (from requests->jupyter-dash) (2025.1.3
1)
Requirement already satisfied: six>=1.7.0 in /opt/anaconda3/lib/pyth
on3.12/site-packages (from retrying->jupyter-dash) (1.16.0)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /opt/anaconda
3/lib/python3.12/site-packages (from jedi>=0.16->ipython->jupyter-da
sh) (0.8.3)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/anaconda3/li
b/python3.12/site-packages (from Jinja2>=3.1.2->flask->jupyter-dash)
(2.1.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/anacon
da3/lib/python3.12/site-packages (from jupyter-client>=6.1.12->ipyke
rnel->jupyter-dash) (2.9.0.post0)
Requirement already satisfied: platformdirs>=2.5 in /opt/anaconda3/l
ib/python3.12/site-packages (from jupyter-core!=5.0.*,>=4.12->ipyker
nel->jupyter-dash) (3.10.0)
Requirement already satisfied: ptyprocess>=0.5 in /opt/anaconda3/li
b/python3.12/site-packages (from pexpect>4.3->ipython->jupyter-dash)
(0.7.0)
Requirement already satisfied: tenacity>=6.2.0 in /opt/anaconda3/li
b/python3.12/site-packages (from plotly>=5.0.0->dash->jupyter-dash)
(8.2.3)
Requirement already satisfied: wcwidth in /opt/anaconda3/lib/python
3.12/site-packages (from prompt-toolkit<3.1.0,>=3.0.41->ipython->jup
yter-dash) (0.2.5)
Requirement already satisfied: zipp>=0.5 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from importlib-metadata->dash->jupyter-dash) (
3.17.0)
Requirement already satisfied: executing in /opt/anaconda3/lib/pytho
n3.12/site-packages (from stack-data->ipython->jupyter-dash) (0.8.3)
Requirement already satisfied: asttokens in /opt/anaconda3/lib/pytho
n3.12/site-packages (from stack-data->ipython->jupyter-dash) (2.0.5)
Requirement already satisfied: pure-eval in /opt/anaconda3/lib/pytho
n3.12/site-packages (from stack-data->ipython->jupyter-dash) (0.2.2)
```

In [257…

```
import os
os.system("lsof -i :8050")  # See what's using the port
os.system("kill $(lsof -t -i :8050)")
```

```
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec
... or kill -l [sigspec]
```

Out[257… 256

In [259…

```
from jupyter_dash import JupyterDash
from dash import dcc, html, Input, Output
import dash_bootstrap_components as dbc
import pandas as pd
import numpy as np
import plotly.express as px

# === Load Data ===
appointments = pd.read_csv("appointments_regional.csv")
duration = pd.read_csv("actual_duration.csv")
categories = pd.read_excel("national_categories.xlsx")
tweets = pd.read_csv("tweets.csv")
icb_lookup = pd.read_csv("icb_locations.csv")  # Ensure this file i
```

```
# === Preprocessing ===
appointments['appointment_month'] = pd.to_datetime(appointments['ap
appointments['year'] = appointments['appointment_month'].dt.year

# KPIs
def compute_kpis(year):
    df = appointments if year == 'All' else appointments[appointmen
    total = df['count_of_appointments'].sum()
    missed = df[df['appointment_status'] == 'DNA']['count_of_appoin
    rate = (missed / total * 100) if total else 0
    return f"{total:,}", f"{missed:,}", f"{rate:.2f}%"

# Regional summary
regional = appointments.dropna(subset=['icb_ons_code', 'count_of_ap
summary_by_icb = regional.groupby('icb_ons_code').agg(
    total_appointments=('count_of_appointments', 'sum'),
    missed_appointments=('count_of_appointments', lambda x: x[regio
).reset_index()

np.random.seed(42)
summary_by_icb['population'] = np.random.randint(500000, 2000000, l
summary_by_icb['missed_rate'] = summary_by_icb['missed_appointments
summary_by_icb['missed_per_1000'] = summary_by_icb['missed_appointm
summary_by_icb['capacity_utilisation'] = summary_by_icb['total_appo
merged = summary_by_icb.merge(icb_lookup[['icb_ons_code', 'icb_ons_
merged['region_label'] = merged['icb_ons_name']

# === Chart Functions ===
def missed_by_region_chart():
    top = merged.sort_values(by='missed_rate', ascending=False).hea
    return px.bar(top, x='missed_rate', y='region_label', orientati

def regional_utilisation_chart():
    top = merged.sort_values(by='capacity_utilisation', ascending=F
    return px.bar(top, x='capacity_utilisation', y='region_label',

def duration_distribution_chart():
    duration_map = {
        '1-5 Minutes': 3, '6-10 Minutes': 8, '11-15 Minutes': 13,
        '16-20 Minutes': 18, '21-30 Minutes': 25, '31-60 Minutes':
    }
    duration['duration_minutes'] = duration['actual_duration'].map(
    clean = duration.dropna(subset=['duration_minutes'])
    grouped = clean.groupby('duration_minutes')['count_of_appointme
    return px.bar(grouped, x='duration_minutes', y='count_of_appoin
                  title="Total Appointments by Duration (Minutes)")

def service_setting_chart():
    grouped = categories.groupby('service_setting')['count_of_appoi
    return px.bar(grouped, x='service_setting', y='count_of_appointm

def hashtag_frequency_chart():
    if 'tweet_entities_hashtags' not in tweets.columns:
        return px.bar(title="No hashtag data found")
```

```
    tweets['tweet_entities_hashtags'] = tweets['tweet_entities_hash
    all_tags = tweets['tweet_entities_hashtags'].str.split(',').exp
    top_tags = all_tags.value_counts().head(10).reset_index()
    top_tags.columns = ['hashtag', 'count']

    return px.bar(top_tags, x='count', y='hashtag', orientation='h'

# === App Layout ===
app = JupyterDash(__name__, external_stylesheets=[dbc.themes.BOOTSTR

year_options = [{'label': 'All', 'value': 'All'}] + \
               [{'label': str(y), 'value': str(y)} for y in sorted(

app.layout = dbc.Container([
    dbc.Row([
        dbc.Col(html.H2("NHS Intelligence Dashboard"), width=8),
        dbc.Col([
            html.Label("Filter by Year:"),
            dcc.Dropdown(id='year-dropdown', options=year_options,
        ], width=4)
    ], align='center', className="mb-3"),

    dcc.Tabs(id='tabs', value='overview', children=[
        dcc.Tab(label='Overview', value='overview'),
        dcc.Tab(label='Missed', value='missed'),
        dcc.Tab(label='Duration', value='duration'),
        dcc.Tab(label='Settings', value='settings'),
        dcc.Tab(label='Sentiment', value='sentiment')
    ], className="mb-4"),

    html.Div(id='tab-content')
], fluid=True)

# === Callbacks ===
@app.callback(
    Output('tab-content', 'children'),
    Input('tabs', 'value'),
    Input('year-dropdown', 'value')
)
def render_tab(tab, selected_year):
    if tab == 'overview':
        total, missed, rate = compute_kpis(selected_year)
        return dbc.Row([
            dbc.Col(dbc.Card(dbc.CardBody([
                html.H5("Total Appointments"), html.H4(total),
                html.Small("All sources", className="text-muted")
            ]), color="light", className="mb-3"), width=4),
            dbc.Col(dbc.Card(dbc.CardBody([
                html.H5("Missed Appointments"), html.H4(missed),
                html.Small("Status: DNA", className="text-muted")
            ]), color="warning", className="mb-3"), width=4),
            dbc.Col(dbc.Card(dbc.CardBody([
                html.H5("Missed Rate"), html.H4(rate),
                html.Small("Percentage missed", className="text-mut
            ]), color="danger", className="mb-3"), width=4),
        ])
```

```
    elif tab == 'missed':
        return html.Div([
            dcc.Graph(figure=missed_by_region_chart()),
            dcc.Graph(figure=regional_utilisation_chart())
        ])
    elif tab == 'duration':
        return html.Div([
            dcc.Graph(figure=duration_distribution_chart())
        ])
    elif tab == 'settings':
        return html.Div([
            dcc.Graph(figure=service_setting_chart())
        ])
    elif tab == 'sentiment':
        return html.Div([
            dcc.Graph(figure=hashtag_frequency_chart())
        ])
    return html.Div("Tab under development")

# === Launch in Notebook ===
app.run(mode='inline', debug=True, port=8060)
```

```
/opt/anaconda3/lib/python3.12/site-packages/dash/dash.py:587: UserWa
rning:

JupyterDash is deprecated, use Dash instead.
See https://dash.plotly.com/dash-in-jupyter for more details.
```

NHS Intelligence Dashboard

Filter by Year:

All

Overview

Missed

Duration

Settings

Sentiment

Total Appointments

742,804,525

All source

Missed Appointments

30,911,233

Status: DNA

Missed Rate

4.16%

Percentage missed

Errors Callbacks v3.0.4 Server

In [260…

```
missed = df[df['appointment_status'] == 'DNA']['count_of_appointmen
total = df[df['appointment_status'].isin(['Attended', 'DNA'])]['cou
missed_rate = (missed / total) * 100
print(f"Missed Appointment Rate: {missed_rate:.2f}%")
```

```
Missed Appointment Rate: 4.69%
```

In [265…

```
import pandas as pd

# Load appointments_regional.csv
df = pd.read_csv("appointments_regional.csv")
df['appointment_month'] = pd.to_datetime(df['appointment_month'], e
df = df[df['appointment_status'].isin(['Attended', 'DNA'])]

# 1. Avg. Monthly Appointments
avg_monthly_appointments = round(df.groupby('appointment_month')['c
```

```
# 2. Missed Appointment Rate (%)
missed = df[df['appointment_status'] == 'DNA']['count_of_appointmen
total = df['count_of_appointments'].sum()
missed_rate = (missed / total) * 100

# 3. Top ICB by appointment volume
top_icb_data = df.groupby('icb_ons_code')['count_of_appointments'].
top_icb_code = top_icb_data.idxmax()
top_icb_appointments = int(top_icb_data.max())

# 4. Face-to-Face Mode (%)
valid_modes = df[df['appointment_mode'].notna()]
face_to_face_total = valid_modes[valid_modes['appointment_mode'] ==
face_to_face_pct = (face_to_face_total / valid_modes['count_of_appo

# 5. General Practice Share (%) from national_categories.xlsx
nc = pd.read_excel("national_categories.xlsx", usecols=["service_se
nc = nc.dropna(subset=['service_setting', 'count_of_appointments'])
gp_total = nc[nc['service_setting'] == 'General Practice']['count_o
overall_total = nc['count_of_appointments'].sum()
gp_share = (gp_total / overall_total) * 100

# Display KPIs
kpi_results = {
    "Avg. Monthly Appointments": avg_monthly_appointments,
    "Missed Appointment Rate (%)": round(missed_rate, 2),
    "Top ICB Code": top_icb_code,
    "Top ICB Appointments": top_icb_appointments,
    "Face-to-Face Mode (%)": round(face_to_face_pct, 2),
    "General Practice Share (%)": round(gp_share, 2)
}

for k, v in kpi_results.items():
    print(f"{k}: {v}")
```

```
Avg. Monthly Appointments: 23622237
Missed Appointment Rate (%): 4.36
Top ICB Code: E54000050
Top ICB Appointments: 41051227
Face-to-Face Mode (%): 58.84
General Practice Share (%): 90.74
```

In [267…

```
import pandas as pd

# Load main dataset
df = pd.read_csv("appointments_regional.csv")
df.columns = df.columns.str.strip()  # Remove any extra whitespace
df['appointment_month'] = pd.to_datetime(df['appointment_month'], e

# Filter to August 2021 – June 2022
df = df[(df['appointment_month'] >= '2021-08-01') & (df['appointmen
df = df[df['appointment_status'].isin(['Attended', 'DNA'])]

# 1. Average Monthly Appointments
avg_monthly_appointments = round(df.groupby('appointment_month')['c

# 2. Missed Appointment Rate (%)
```

```
missed = df[df['appointment_status'] == 'DNA']['count_of_appointmen
total = df['count_of_appointments'].sum()
missed_rate = (missed / total) * 100

# 3. Top & Bottom ICBs by appointment volume (by name)
top_icb_group = df.groupby('icb_ons_code')['count_of_appointments']
top_icb_name = top_icb_group.idxmax()
top_icb_appointments = int(top_icb_group.max())
bottom_icb_name = top_icb_group.idxmin()
bottom_icb_appointments = int(top_icb_group.min())

# 4. Face-to-Face Mode (%)
valid_modes = df[df['appointment_mode'].notna()]
face_to_face_total = valid_modes[valid_modes['appointment_mode'] ==
face_to_face_pct = (face_to_face_total / valid_modes['count_of_appo

# 5. General Practice Share (%) — from national_categories.xlsx
nc = pd.read_excel("national_categories.xlsx", usecols=["appointmen
nc['appointment_month'] = pd.to_datetime(nc['appointment_month'], e
nc = nc[(nc['appointment_month'] >= '2021-08-01') & (nc['appointmen
nc = nc.dropna(subset=['service_setting', 'count_of_appointments'])
gp_total = nc[nc['service_setting'] == 'General Practice']['count_o
overall_total = nc['count_of_appointments'].sum()
gp_share = (gp_total / overall_total) * 100

# 6. Peak Month
monthly_totals = df.groupby('appointment_month')['count_of_appointm
peak_month = monthly_totals.idxmax().strftime('%B %Y')
peak_volume = int(monthly_totals.max())

# Final KPI Summary
print(" KPI Summary (Aug 2021 – Jun 2022):")
print(f"1. Avg. Monthly Appointments: {avg_monthly_appointments:,}"
print(f"2. Missed Appointment Rate (%): {missed_rate:.2f}%")
print(f"3. Top ICB: {top_icb_name} — {top_icb_appointments:,} appoi
print(f"4. Bottom ICB: {bottom_icb_name} — {bottom_icb_appointments
print(f"5. Face-to-Face Mode (%): {face_to_face_pct:.2f}%")
print(f"6. General Practice Share (%): {gp_share:.2f}%")
print(f"7. Peak Month: {peak_month} — {peak_volume:,} appointments"
```

```
 KPI Summary (Aug 2021 – Jun 2022):
1. Avg. Monthly Appointments: 25,816,007
2. Missed Appointment Rate (%): 4.69%
3. Top ICB: E54000050 — 16,230,572 appointments
4. Bottom ICB: E54000011 — 2,466,146 appointments
5. Face-to-Face Mode (%): 61.73%
6. General Practice Share (%): 90.74%
7. Peak Month: November 2021 — 29,095,752 appointments
```

In [269…

```
import pandas as pd

# Load data
df = pd.read_csv("appointments_regional.csv")
pop_df = pd.read_csv("icb_locations.csv")

# Clean columns
df.columns = df.columns.str.strip()
```

```
pop_df.columns = pop_df.columns.str.strip()

# Filter dates
df['appointment_month'] = pd.to_datetime(df['appointment_month'], e
df = df[(df['appointment_month'] >= '2021-08-01') & (df['appointmen

# Appointments by ICB
appointments_by_icb = df.groupby('icb_ons_code')['count_of_appointm

# Merge with population
merged = pd.merge(
    appointments_by_icb,
    pop_df[['icb_ons_code', 'icb_ons_name', 'icb_registered_populat
    on='icb_ons_code',
    how='left'
)

# Calculate per 1,000 people
merged['appointments_per_1000'] = (merged['count_of_appointments']

# Top and bottom
top = merged.sort_values(by='appointments_per_1000', ascending=Fals
bottom = merged.sort_values(by='appointments_per_1000', ascending=T

# Show results
print(" Highest Appointments per 1,000:")
print(f"{top['icb_ons_name']} — {top['appointments_per_1000']:.0f}

print("Lowest Appointments per 1,000:")
print(f"{bottom['icb_ons_name']} — {bottom['appointments_per_1000']
```

```
 Highest Appointments per 1,000:
NHS Cornwall and the Isles of Scilly Integrated Care Board — 6201 pe
r 1,000 people

Lowest Appointments per 1,000:
NHS South East London Integrated Care Board — 3850 per 1,000 people
```

In [271…

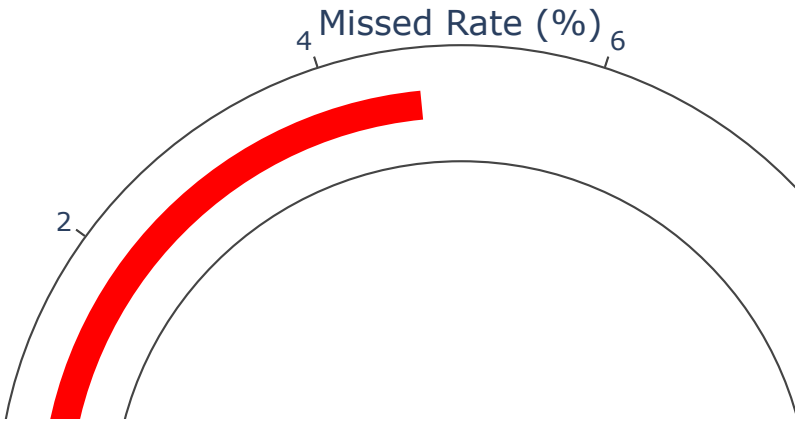
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# KPI values
avg_monthly_appointments = 25816007
peak_volume = 29095752
gp_share = 90.74
peak_utilisation = 115.2

# Slide 1: Utilisation & Capacity
fig1 = make_subplots(
    rows=2, cols=2,
    specs=[[{"type": "indicator"}, {"type": "indicator"}],
           [{"type": "indicator"}, {"type": "indicator"}]],
    subplot_titles=[
        "Monthly Avg.", "Peak Month: November 2021",
        "GP Share (%)", "Utilisation (%)"
    ]
)
```

```
fig1.add_trace(go.Indicator(mode="number", value=avg_monthly_appoin
                            number={'valueformat': ",", 'font': {'s

fig1.add_trace(go.Indicator(mode="number", value=peak_volume,
                            number={'valueformat': ",", 'font': {'s

fig1.add_trace(go.Indicator(mode="gauge+number", value=gp_share,
                            gauge={'axis': {'range': [0, 100]}, 'ba
                            domain={'x': [0, 1], 'y': [0, 0.75]}, nu

fig1.add_trace(go.Indicator(mode="gauge+number", value=peak_utilisa
                            gauge={'axis': {'range': [0, 140]}, 'ba
                            domain={'x': [0, 1], 'y': [0, 0.75]}, nu

fig1.update_layout(
    title_text="NHS Service Utilisation & Capacity",
    title_font_size=22,
    title_x=0.5,
    font=dict(size=12),
    height=700,
    template="plotly_white",
    margin=dict(t=80, b=40)
)

fig1.show()
```

In [273…

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# KPI values
missed_rate = 4.69
face_to_face_pct = 61.73
top_per_1000 = 6201
bottom_per_1000 = 3850

# Slide 2: Access & Inequality
fig2 = make_subplots(
    rows=2, cols=2,
    specs=[[{"type": "indicator"}, {"type": "indicator"}],
           [{"type": "indicator"}, {"type": "indicator"}]],
```

```
    subplot_titles=[
        "Missed Rate (%)", "Face-to-Face (%)",
        "Top ICB per 1,000", "Lowest ICB per 1,000"
    ]
)

fig2.add_trace(go.Indicator(mode="gauge+number", value=missed_rate,
                            gauge={'axis': {'range': [0, 10]}, 'bar
                            domain={'x': [0, 1], 'y': [0, 0.75]}, nu

fig2.add_trace(go.Indicator(mode="gauge+number", value=face_to_face_
                            gauge={'axis': {'range': [0, 100]}, 'ba
                            domain={'x': [0, 1], 'y': [0, 0.75]}, nu

fig2.add_trace(go.Indicator(mode="gauge+number", value=top_per_1000
                            gauge={'axis': {'range': [3000, 6500]},
                            domain={'x': [0, 1], 'y': [0, 0.75]}, nu

fig2.add_trace(go.Indicator(mode="gauge+number", value=bottom_per_1(
                            gauge={'axis': {'range': [3000, 6500]},
                            domain={'x': [0, 1], 'y': [0, 0.75]}, nu

fig2.update_layout(
    title_text="Access & Inequality Across ICBs",
    title_font_size=22,
    title_x=0.5,
    font=dict(size=12),
    height=700,
    template="plotly_white",
    margin=dict(t=80, b=40)
)

fig2.show()
```



In [ ]: