Hospital Data Analysis Report

Introduction

This report presents an analysis of hospital data, focusing on key aspects such as hospital types, geographical distribution, sector classification (public vs. private), and organizational structures. Understanding these factors is crucial for healthcare planning, resource allocation, and gaining insights into the UK's healthcare landscape. The analysis employs a combination of data processing, statistical analysis, and visualization techniques to provide a comprehensive overview of the dataset.

Objectives

The primary objectives of this analysis are to:

- Clean and preprocess the raw hospital data to ensure data quality and consistency.
- Explore the distribution of hospitals by type and subtype to understand the composition of healthcare facilities.
- Visualize the geographical distribution of hospitals to identify regional patterns and potential disparities.
- Determine the distribution of hospitals between the public (NHS Sector) and private (Independent Sector).
- Analyze the influence of parent organizations on hospital distribution.
- Assess the completeness of contact information for hospitals.
- Examine the density of hospitals across latitude and longitude coordinates.

Methodology

The analysis was conducted using a multi-stage approach, incorporating the following tools and techniques:

1. **Data Cleaning and Preprocessing:**
    - **Tool:** Python (Jupyter Notebook)

- The raw data was loaded and cleaned using the pandas library in Python. This involved handling missing values (imputation or removal), removing duplicates, and standardizing data formats.

2. **Exploratory Data Analysis (EDA):**
    - **Tool:** Python (Jupyter Notebook)
    - Descriptive statistics and data visualization techniques were employed to explore the data's characteristics. Libraries such as matplotlib and seaborn in Python were used to generate plots illustrating the distribution of hospital types, subtypes, sector, and geographical location.
3. **Advanced Data Analysis:**
    - **Tool:** Python (Jupyter Notebook) and Excel
    - Further analysis was performed to aggregate data (e.g., hospitals per city/county), calculate percentage distributions, and analyze parent organization influence. Data was exported to Excel for additional manipulation where necessary.
4. **Data Visualization and Dashboarding:**
    - **Tool:** Power BI
    - Key findings from the analysis were used to create interactive dashboards in Power BI. These dashboards provide a user-friendly interface for exploring the data and gaining actionable insights.

Analysis

General Information

- **OrganisationID**: Unique numeric identifier for each organization.
- **OrganisationCode**: Alphanumeric code assigned to each organization.
- **OrganisationType / SubType**: Classifications, generally both marked as “Hospital”.
- **Sector**: Indicates the sector (e.g., "NHS Sector", "Independent Sector").
- **OrganisationStatus**: Status of the organization (e.g., "Visible").

Administrative & Location Details

- **IsPimsManaged**: Boolean indicating whether the organization is managed by the PIMS system.
- **OrganisationName**: Full name of the hospital or organization.
- **Address1, Address2, Address3, City, County, Postcode**: Address components.
- **Latitude / Longitude**: Geographical coordinates.

Parent Organization

- **ParentODSCode / ParentName**: Code and name of the parent organization.

Contact Information

- **Phone / Email / Website / Fax,,,**: Contact details (note: "Fax,,," has an odd column name with trailing commas).

1. Data Cleaning & Preprocessing

This stage focuses on preparing the raw data for analysis. Real-world data is often incomplete, inconsistent, or contains errors. Data cleaning ensures the data's quality, accuracy, and consistency, which are crucial for generating reliable insights. Preprocessing involves transforming the data into a suitable format for analysis, such as handling missing values or standardizing formats.

1.1 Import Libraries and Load Data

We begin by importing the necessary Python libraries (pandas for data manipulation, numpy for numerical operations, matplotlib and seaborn for visualization). The dataset is then loaded into a pandas DataFrame, which is a table-like structure. We then inspect the data's structure (df.info()) and the first few rows (df.head()) to understand its contents.

Python Code

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
file_path = "data/Hospital.csv"  # Adjust the path if needed
df = pd.read_csv(file_path)

# Display basic info
df.info()
df.head()
```

Result

```
<class 'pandas.core.frame.DataFrame'>
Index: 1211 entries, 0 to 1210
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   OrganisationID      1211 non-null   int64
 1   OrganisationCode    1211 non-null   object
 2   OrganisationType    1211 non-null   object
 3   SubType             1211 non-null   object
 4   Sector              1211 non-null   object
 5   OrganisationStatus  1211 non-null   object
 6   IsPimsManaged       1211 non-null   bool
 7   OrganisationName    1211 non-null   object
 8   Address1            883 non-null    object
 9   Address2            727 non-null    object
 10  Address3            147 non-null    object
 11  City                1196 non-null   object
 12  County              973 non-null    object
 13  Postcode            1210 non-null   object
 14  Latitude            1209 non-null   float64
 15  Longitude           1209 non-null   float64
 16  ParentODSCode       1211 non-null   object
 17  ParentName          1211 non-null   object
 18  Phone               961 non-null    object
 19  Email               422 non-null    object
 20  Website             853 non-null    object
 21  Fax,,,              1209 non-null   object
dtypes: bool(1), float64(2), int64(1), object(18)
memory usage: 209.3+ KB
```

	OrganisationID	OrganisationCode	OrganisationType	SubType	Sector	OrganisationStatus
0	17970	NDA07	Hospital	Hospital	Independent Sector	Visible
1	17981	NDA18	Hospital	Hospital	Independent Sector	Visible
2	18102	NLT02	Hospital	Hospital	NHS Sector	Visible
3	18138	NMP01	Hospital	Hospital	Independent Sector	Visible
4	18142	NMV01	Hospital	Hospital	Independent Sector	Visible

5 rows × 22 columns

1.2. Check for Missing Values

Missing values can significantly skew analysis results. We identify the extent of missing data in each column to determine appropriate handling strategies. This step helps us understand the completeness of the dataset and potential biases.

Python Code

```
# Count missing values per column
missing_values = df.isnull().sum().sort_values(ascending=False)
missing_percentage = (missing_values / len(df)) * 100

# Display missing values
missing_data = pd.DataFrame({'Missing Values': missing_values, 'Percentage': missing_percentage})
print(missing_data[missing_data['Missing Values'] > 0])

#number of ,,, in Fax,,,
print(df['Fax,,,'].str.strip().str.cat(sep='').count(',,,'))
```

Result

	Missing Values	Percentage
Address3	1064	87.861272
Email	789	65.152766
Address2	484	39.966969
Website	358	29.562345
Address1	328	27.085054
Phone	250	20.644096
County	238	19.653179
City	15	1.238646
Longitude	2	0.165153
Latitude	2	0.165153
Fax,,,	2	0.165153
Postcode	1	0.082576

1154

1.3 Handle Missing Values¶

Based on the missing value analysis, we apply suitable techniques. Columns with excessive missing data may be removed, as they offer limited information. Missing values in categorical columns might be replaced with a placeholder like 'Unknown,' while numerical missing values could be imputed using measures of central tendency, such as the median.

Python code

```
# Drop columns with excessive missing values
columns_to_drop = ['Address3', 'Fax,,,']
df.drop(columns=columns_to_drop, inplace=True)

# Fill missing values for categorical data with 'Unknown'
categorical_columns = ['Address1', 'Address2', 'City', 'County']
df[categorical_columns] = df[categorical_columns].fillna('Unknown')

# Fill missing numerical values with median
df['Latitude'].fillna(df['Latitude'].median(), inplace=True)
df['Longitude'].fillna(df['Longitude'].median(), inplace=True)
```

1.4 Check for Duplicates and Remove Them

Duplicate records can distort analysis, leading to inflated counts or incorrect statistics. We identify and remove duplicate rows to ensure each record represents a unique entity.

Python Code

```python
# Count and remove duplicate rows
duplicates = df.duplicated().sum()
df = df.drop_duplicates()

print(f"Removed {duplicates} duplicate rows.")
```

```
Removed 0 duplicate rows.
```

1.5 Standardize Phone and Website Formats

Inconsistent formatting can hinder analysis. We standardize phone numbers (e.g., removing spaces) and website URLs to ensure uniformity and facilitate accurate comparisons or filtering.

Python Code

```python
# Remove spaces and standardize phone numbers
df['Phone'] = df['Phone'].str.replace(r'\s+', '', regex=True)

# Ensure website URLs are properly formatted
df['Website'] = df['Website'].apply(lambda x: x if isinstance(x, str) and x.startswith('http') else 'Unknown')
```

1.6 Save the cleanned data

The cleaned and preprocessed data is saved to a new file. This ensures that the original data remains unchanged and that subsequent analyses are performed on a consistent and reliable dataset.

Python Code

```
text_columns = df.columns

# Save the cleaned data
cleaned_file_path = "Cleaned_Hospital.csv"
df.to_csv(cleaned_file_path, index=False)
```

2. Exploratory Data Analysis (EDA)

2.1 Distribution of Hospital Types

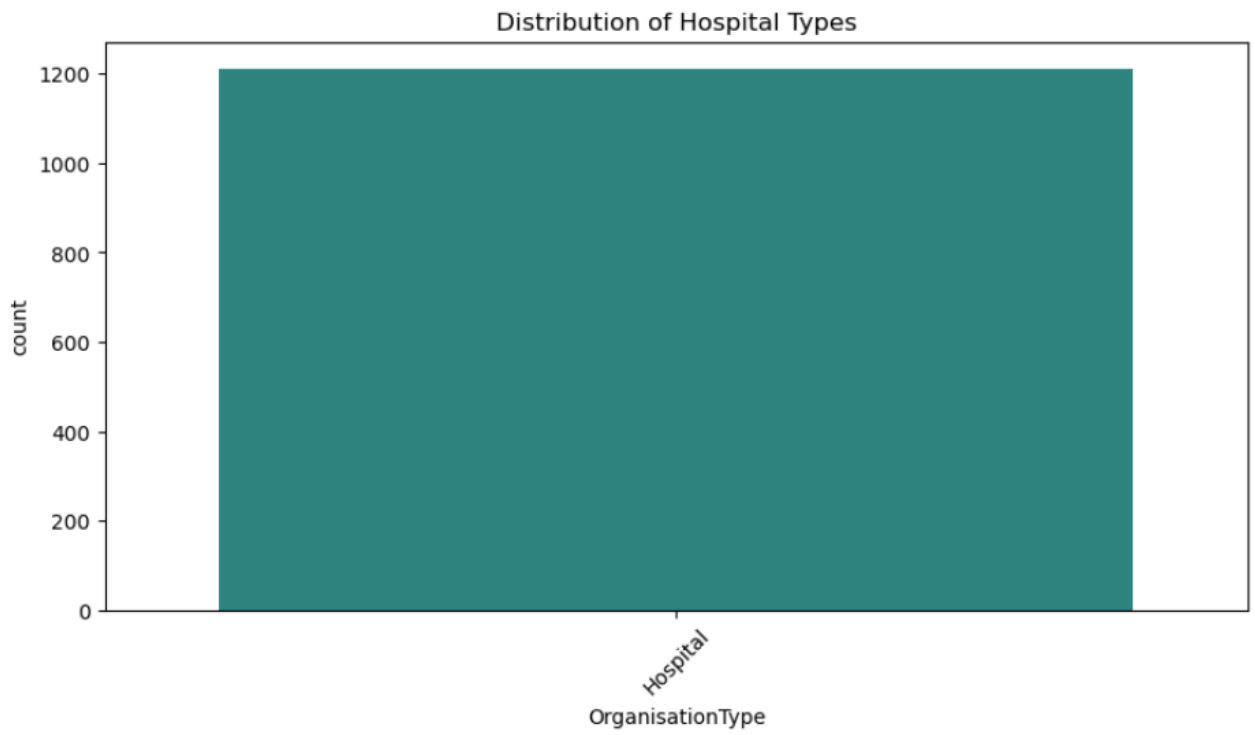
2.1.1 Organization type

We visualize the frequency of different hospital types using a count plot. This helps us understand the composition of our dataset and identify the most common types of hospitals.

Python Code

```
plt.figure(figsize=(10, 5))
sns.countplot(data=df, x='OrganisationType', order=df['OrganisationType'].value_counts().index, palette='viridis')
plt.xticks(rotation=45)
plt.title("Distribution of Hospital Types")
plt.show()
```

Result



Interpretation:

The count plot clearly shows that 'Hospital' is the predominant OrganisationType. This indicates that the dataset primarily focuses on general hospitals rather than other organizational types. This dominance should be considered in further analyses, as it might skew results if not accounted for."

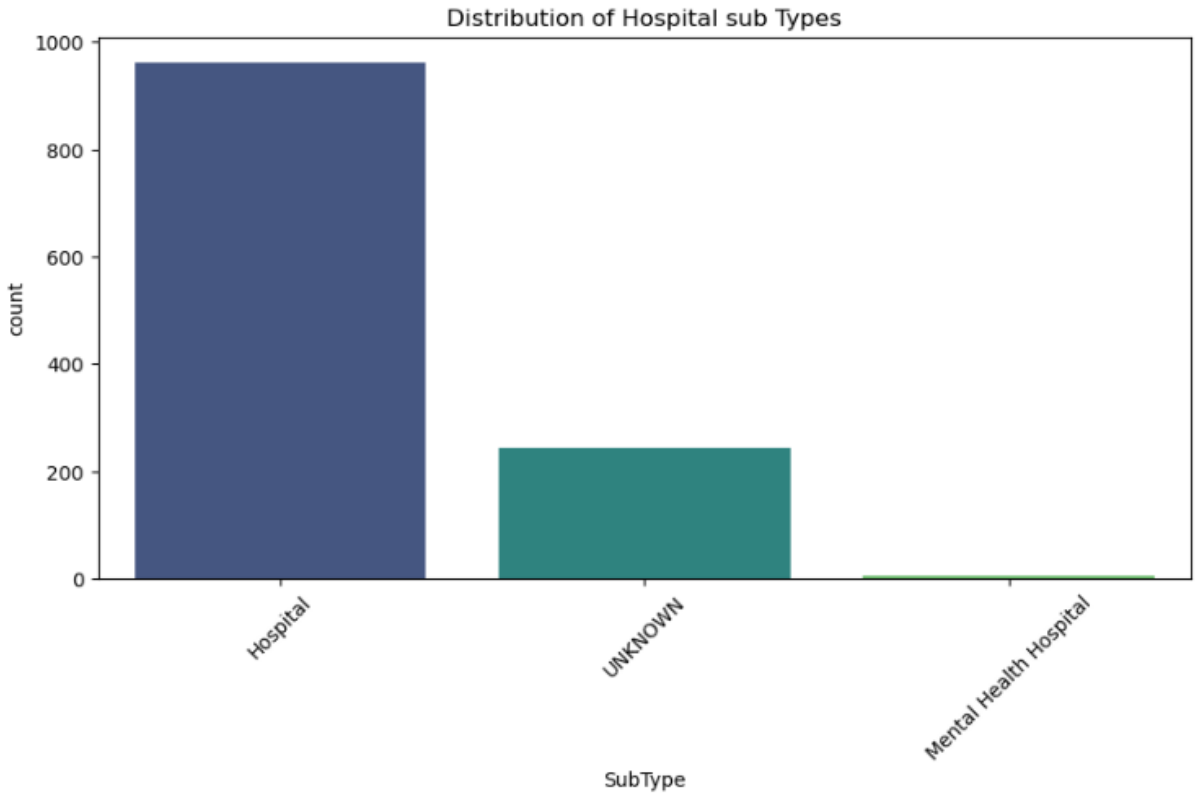
2.1.1 Organization Subtype

We visualize the frequency of different hospital sub-types using a count plot. This helps us understand the composition of sub types of hospitals in our dataset and identify the most common sub types of hospitals.

Python Code

```
plt.figure(figsize=(10, 5))
sns.countplot(data=df, x='OrganisationType', order=df['OrganisationType'].value_counts().index, palette='viridis')
plt.xticks(rotation=45)
plt.title("Distribution of Hospital Types")
plt.show()
```

Result



- □

Interpretation:

"The count plot shows that 'Hospital' is the predominant SubType, followed by 'UNKNOWN' and then 'Mental Health Hospital'. This indicates that the dataset primarily focuses on general hospitals rather than other sub types. It also indicates that there are many entries where the sub type is unknown. This should be considered in further analyses, as it might skew results if not accounted for."

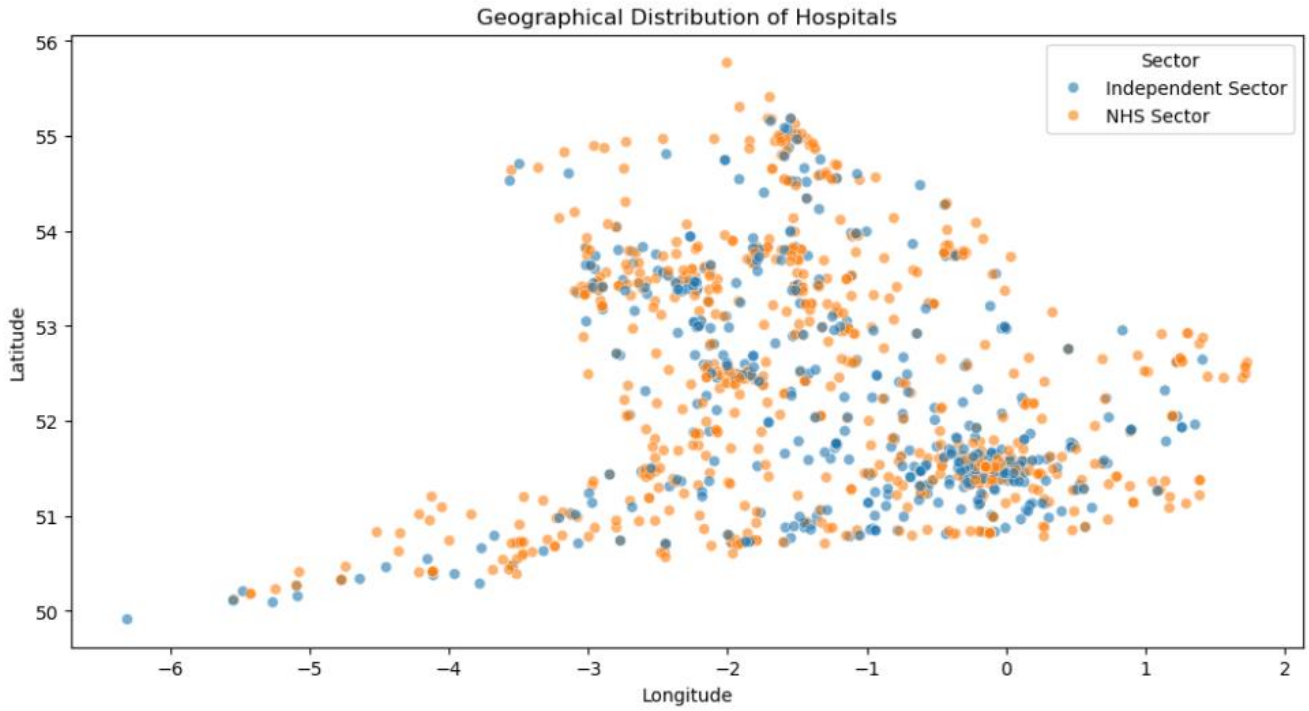
2.2 Geographic Distribution of Hospitals

A scatter plot of latitude and longitude is used to visualize the geographical distribution of hospitals. Color-coding by sector (public or private) adds another dimension to the analysis, revealing potential spatial patterns related to hospital ownership.

Python Code

```
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='Longitude', y='Latitude', hue='Sector', alpha=0.6)
plt.title("Geographical Distribution of Hospitals")
plt.show()
```

Result



Interpretation:

"The scatter plot reveals a concentration of hospitals in certain areas, particularly in the southern regions of the UK. Both NHS Sector and Independent Sector hospitals are distributed throughout the region, but there might be local clusters of one sector or the other. Further analysis could explore the reasons behind these clusters and their implications for healthcare access."

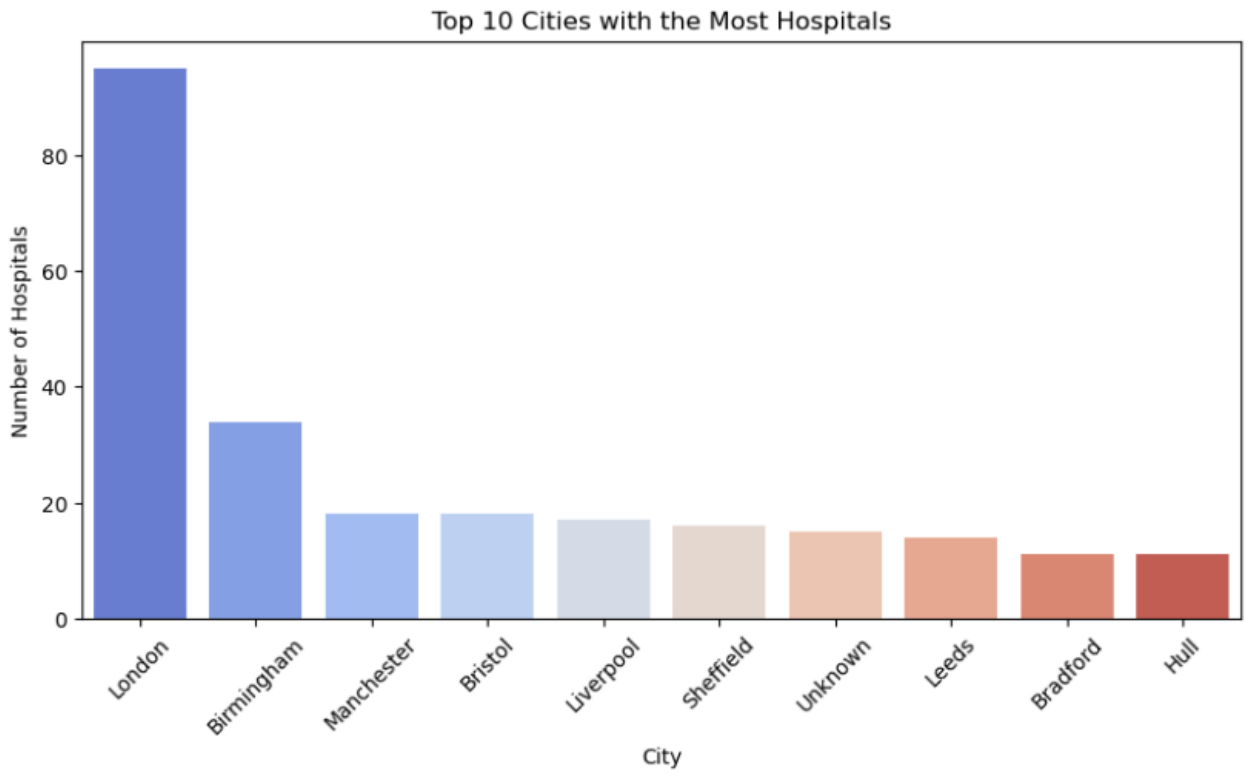
2.3 Top 10 Cities with the Most Hospitals

We identify and visualize the top 10 cities with the highest number of hospitals using a bar plot. This helps to pinpoint areas with high concentrations of healthcare facilities, which could be relevant for resource planning or competitive analysis.

Python Code

```
top_cities = df['City'].value_counts().nlargest(10)

plt.figure(figsize=(10, 5))
sns.barplot(x=top_cities.index, y=top_cities.values, palette="coolwarm")
plt.xticks(rotation=45)
plt.title("Top 10 Cities with the Most Hospitals")
plt.ylabel("Number of Hospitals")
plt.show()
```

Result



- □

Interpretation:

"The bar plot clearly shows that London has a significantly higher number of hospitals compared to other cities. This concentration of hospitals in London likely reflects its larger population and status as a major metropolitan area. Other major cities like Birmingham, Manchester, and Bristol also have a notable number of hospitals."

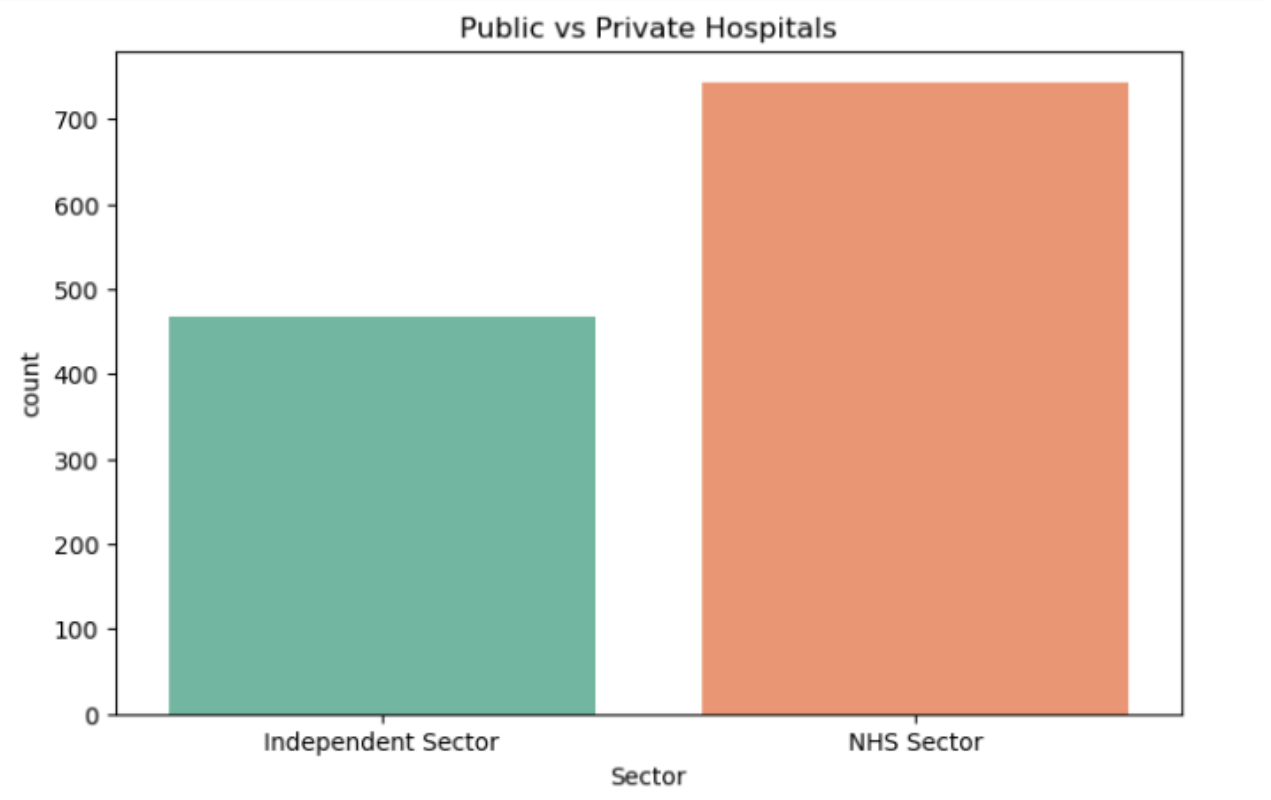
2.4 Public vs Private Hospitals

A count plot is used to compare the number of hospitals in the public (NHS Sector) and private (Independent Sector). This provides insights into the balance of healthcare provision within the dataset.

Python Code

```
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='Sector', palette='Set2')
plt.title("Public vs Private Hospitals")
plt.show()
```

Result



Interpretation:

"The count plot indicates a higher number of hospitals in the NHS Sector compared to the Independent Sector. This suggests that public healthcare providers form a larger part of the dataset, which aligns with the general structure of healthcare in the UK."

3. Advanced Data Analysis

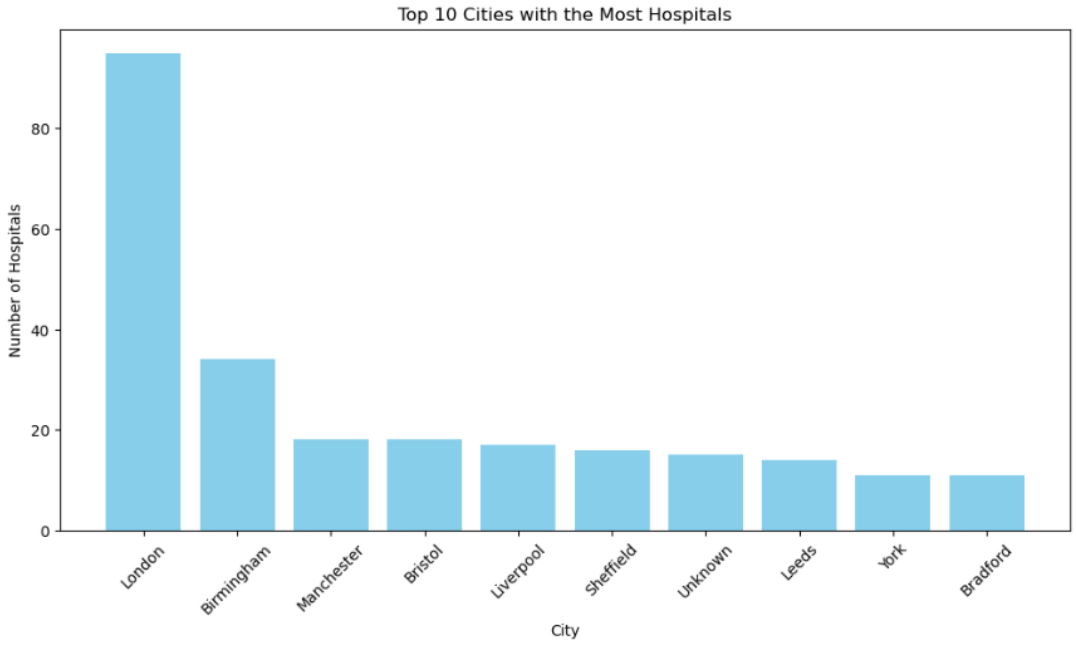
3.1 Number of Hospitals by Region (City, County)

We aggregate the data to count the number of hospitals within each city and county. This aggregated data is useful for regional analysis and can be exported for use in other business intelligence tools. Visualizing the top 10 cities by hospital count using a bar chart provides a clear comparison.

Python Code

```
# Aggregate data for Power BI
hospitals_by_city = df.groupby('City').size().reset_index(name='Hospital Count')
hospitals_by_county = df.groupby('County').size().reset_index(name='Hospital Count')

# Save for Power BI
hospitals_by_city.to_csv("hospitals_by_city.csv", index=False)
hospitals_by_county.to_csv("hospitals_by_county.csv", index=False)

# Optional: Matplotlib validation
plt.figure(figsize=(12, 6))
top_cities = hospitals_by_city.sort_values(by="Hospital Count", ascending=False).head(10)
plt.bar(top_cities['City'], top_cities['Hospital Count'], color='skyblue')
plt.xticks(rotation=45)
plt.title("Top 10 Cities with the Most Hospitals")
plt.xlabel("City")
plt.ylabel("Number of Hospitals")
plt.show()
```

Result



- □

Interpretation:

"The bar chart confirms that London has the highest count, and the visualization makes the differences between the top cities more apparent. This detailed view allows for a more precise comparison of hospital distribution among major cities."

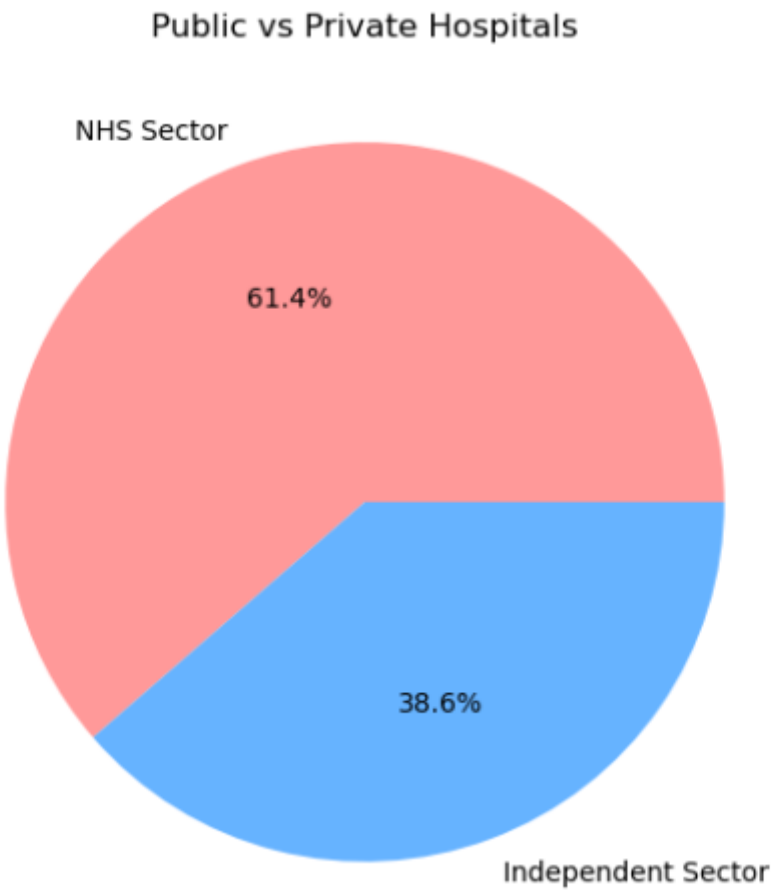
3.2 Percentage of Hospitals Managed by NHS vs Independent Sector

We calculate the percentage distribution of hospitals between the NHS and Independent sectors. This provides a normalized view of sector representation, making it easier to compare proportions regardless of the total number of hospitals. A pie chart effectively visualizes this percentage distribution.

Python Code

```
# Aggregation for Power BI
sector_distribution = df['Sector'].value_counts(normalize=True).reset_index()
sector_distribution.columns = ['Sector', 'Percentage']

# Save for Power BI
sector_distribution.to_csv("sector_distribution.csv", index=False)

# Optional: Matplotlib validation
plt.figure(figsize=(6, 6))
plt.pie(sector_distribution['Percentage'], labels=sector_distribution['Sector'], autopct='%1.1f%%', colors=['#ff9999','#66b3ff'])
plt.title("Public vs Private Hospitals")
plt.show()
```

Result



Interpretation:

"The pie chart clearly illustrates that the NHS Sector represents a larger proportion (61.4%) of hospitals compared to the Independent Sector (38.6%). This visual representation emphasizes the dominance of public healthcare provision in the dataset."

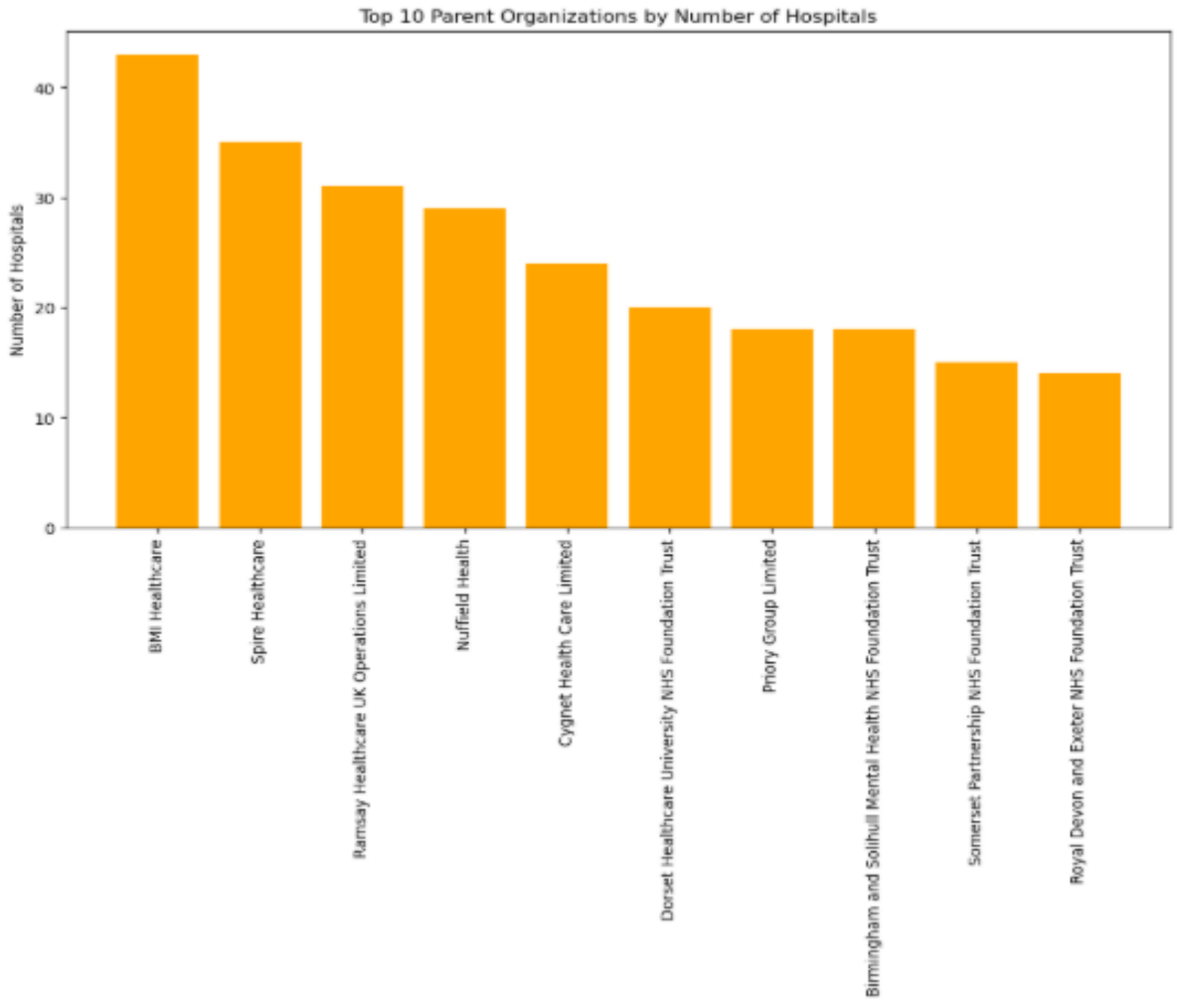
3.3 Parent Organization Influence on Hospital Distribution

We analyse the distribution of hospitals across parent organizations to understand the influence of major healthcare providers. Counting hospitals per parent organization and visualizing the top 10 allows us to identify key players in the healthcare landscape.

Python Code

```python
# Count hospitals per parent organization
parent_distribution = df.groupby('ParentName').size().reset_index(name='Hospital Count')

# Save for Power BI
parent_distribution.to_csv("parent_distribution.csv", index=False)

# Optional: Matplotlib validation
plt.figure(figsize=(12, 6))
top_parents = parent_distribution.sort_values(by="Hospital Count", ascending=False).head(10)
plt.bar(top_parents['ParentName'], top_parents['Hospital Count'], color='orange')
plt.xticks(rotation=90)
plt.title("Top 10 Parent Organizations by Number of Hospitals")
plt.ylabel("Number of Hospitals")
plt.show()
```

Result



Interpretation:

"The bar chart highlights the significant influence of several large parent organizations, such as BMI Healthcare and Spire Healthcare, in terms of the number of hospitals. This suggests a degree of consolidation in the healthcare market, with these major players having a substantial impact on healthcare delivery."

3.4 Completeness of Contact Information

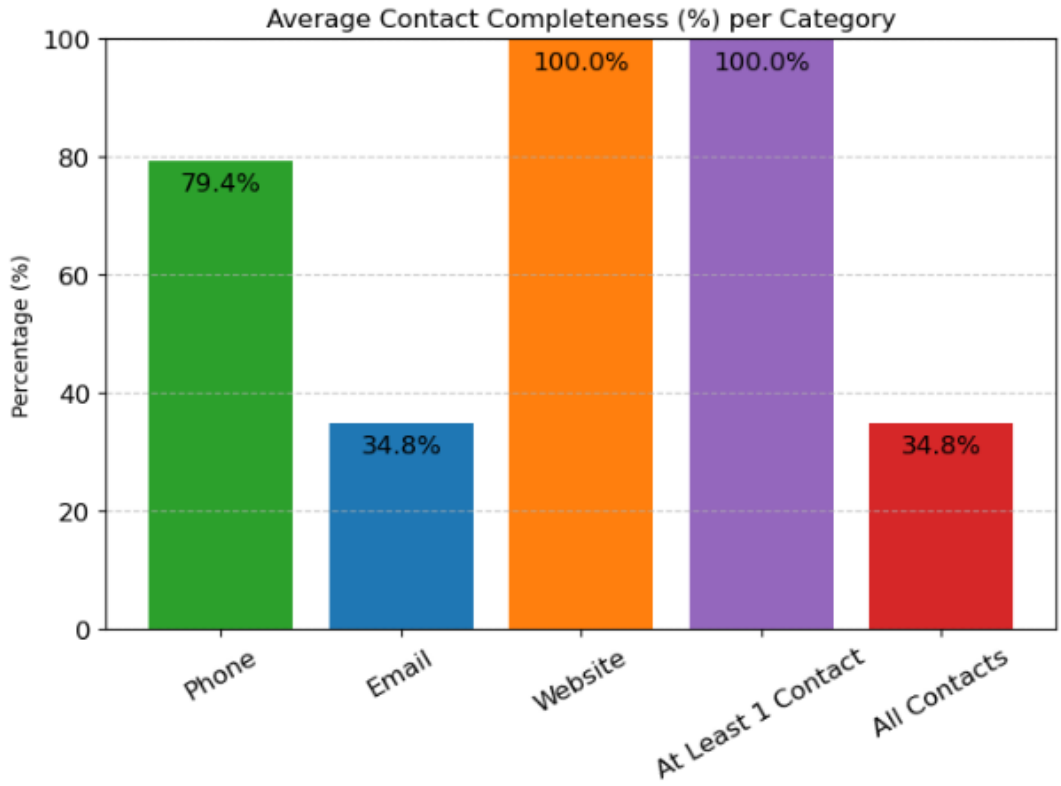
This analysis aims to assess the completeness of contact information provided for each hospital. Complete and accurate contact details (phone, email, website) are essential for communication, referrals, and public accessibility. Evaluating the availability of this information helps identify potential gaps and areas where data maintenance is needed.

Python Code

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

# Define completeness per OrganisationCode
contact_completeness = df[['OrganisationCode', 'Phone', 'Email', 'Website']].copy()

# Convert to boolean (1 if not null, 0 if null)
contact_completeness['Has_Phone'] = contact_completeness['Phone'].notna().astype(int)
contact_completeness['Has_Email'] = contact_completeness['Email'].notna().astype(int)
contact_completeness['Has_Website'] = contact_completeness['Website'].notna().astype(int)

# Calculate individual completeness percentages
contact_completeness['Phone Completeness (%)'] = contact_completeness['Has_Phone'] * 100
contact_completeness['Email Completeness (%)'] = contact_completeness['Has_Email'] * 100
contact_completeness['Website Completeness (%)'] = contact_completeness['Has_Website'] * 100

# Calculate completeness for at least 1 contact
contact_completeness['At Least 1 Contact (%)'] = contact_completeness[['Has_Phone', 'Has_Email', 'Has_Website']].max(axis=1) * 100

# Calculate completeness for all contacts available
contact_completeness['All Contacts (%)'] = contact_completeness[['Has_Phone', 'Has_Email', 'Has_Website']].min(axis=1) * 100

# Keep only relevant columns
contact_completeness = contact_completeness[['OrganisationCode',
                                             'Phone Completeness (%)',
                                             'Email Completeness (%)',
                                             'Website Completeness (%)',
                                             'At Least 1 Contact (%)',
                                             'All Contacts (%)']].drop_duplicates()
```

```python
# Save for Power BI
contact_completeness.to_csv("contact_completeness.csv", index=False)

# Matplotlib Visualization: Stacked Bar Chart
plt.figure(figsize=(8, 5))
categories = ['Phone', 'Email', 'Website', 'At Least 1 Contact', 'All Contacts']
percentages = [
    contact_completeness['Phone Completeness (%)'].mean(),
    contact_completeness['Email Completeness (%)'].mean(),
    contact_completeness['Website Completeness (%)'].mean(),
    contact_completeness['At Least 1 Contact (%)'].mean(),
    contact_completeness['All Contacts (%)'].mean()
]

bars = plt.bar(categories, percentages, color=['#2ca02c', '#1f77b4', '#ff7f0e', '#9467bd', '#d62728'])

# Add percentage labels on top of bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval-5, f"{yval:.1f}%", ha='center', fontsize=12)

plt.title("Average Contact Completeness (%) per Category")
plt.ylabel("Percentage (%)")
plt.ylim(0, 100)
plt.xticks(rotation=30, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Result



3.4 Completeness of Contact Information

- **Interpretation:**
    - **Phone: Approximately 79.4% of hospitals have phone numbers listed. This relatively high percentage indicates good availability of phone contact information.**
    - **Email: Only 34.8% of hospitals have email addresses listed. This low percentage suggests a significant gap in the availability of email contact information.**
    - **Website: 100% of hospitals have website URLs listed. This indicates excellent availability of online contact information.**
    - **At least 1 contact: 100% of hospitals have at least one form of contact listed (phone, email, or website). This ensures that all hospitals are reachable in some way.**
    - **All contacts: Only 34.8% of hospitals have all forms of contact listed (phone, email, and website). This highlights the significant deficiency in the availability of email contact information.**

In summary:

- **Website availability is excellent.**
- **Phone number availability is good.**
- **Email address availability is very poor.**
- **All hospitals have at least one form of contact, but most lack a complete set of contact details.**

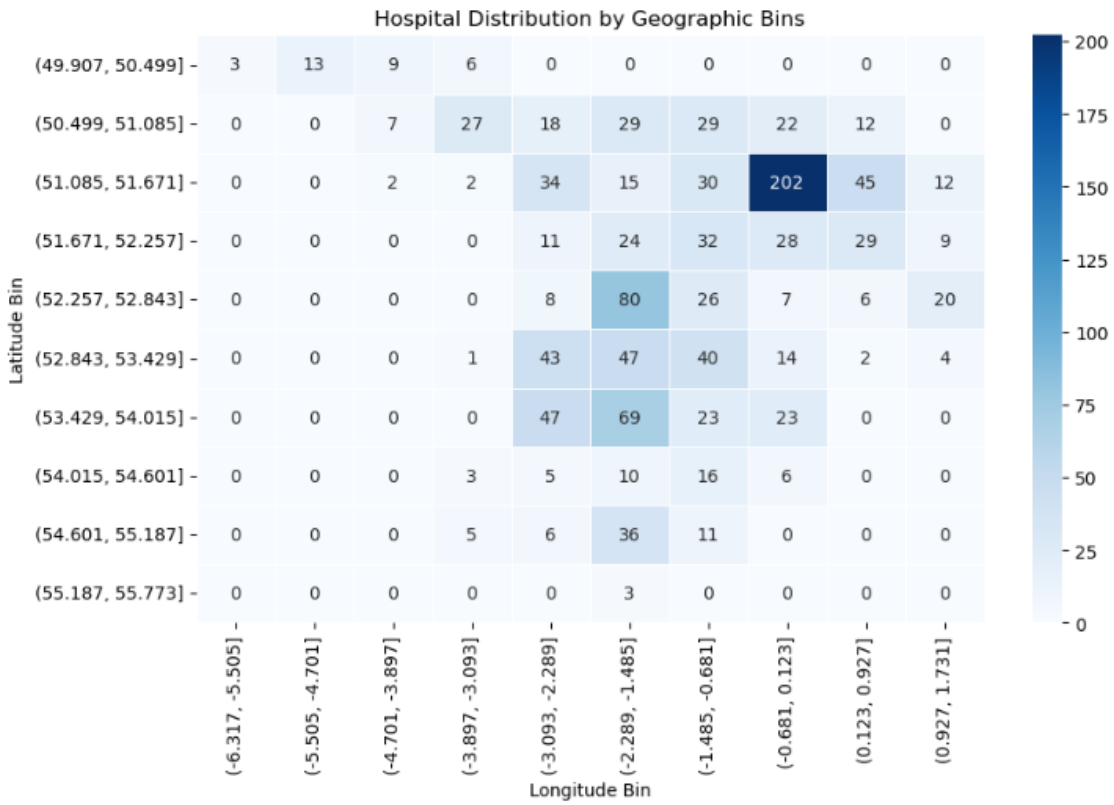
This information is crucial for assessing the accessibility of hospitals and identifying areas where contact information needs improvement.

3.5 Geographic Coverage - Number of Hospitals per Latitude/Longitude

Python Code

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

# Binning latitude and longitude to create regional clusters
df['Latitude_bin'] = pd.cut(df['Latitude'], bins=10)
df['Longitude_bin'] = pd.cut(df['Longitude'], bins=10)

# Aggregate data
geo_distribution = df.groupby(['Latitude_bin', 'Longitude_bin'], observed=False).size().reset_index(name='Hospital Count')

# Save for Power BI
geo_distribution.to_csv("geo_distribution.csv", index=False)

# Optional: Matplotlib validation (Heatmap)
plt.figure(figsize=(10, 6))

# Fix pivot table syntax
pivot_table = geo_distribution.pivot(index="Latitude_bin", columns="Longitude_bin", values="Hospital Count")

# Handle missing values by filling NaNs with 0
pivot_table = pivot_table.fillna(0)

# Plot heatmap
sns.heatmap(pivot_table, cmap="Blues", annot=True, fmt=".0f", linewidths=0.5)
plt.title("Hospital Distribution by Geographic Bins")
plt.xlabel("Longitude Bin")
plt.ylabel("Latitude Bin")
plt.show()
```

Result



- **What We Do:**
    - We've created a heatmap that visualizes the distribution of hospitals across geographic regions.
    - The data has been binned into ranges of latitude and longitude, creating a grid.
    - The heatmap's color intensity represents the number of hospitals within each latitude/longitude bin.
    - This allows us to quickly identify areas with high or low hospital density.
- **Why We Do It:**
    - To understand the spatial distribution of hospitals and identify areas with high or low concentration.
    - To reveal potential disparities in healthcare access based on geographic location.
    - To inform healthcare planning and resource allocation decisions.
    - To visually represent where the hospitals are located in a 2d format.

- **Interpretation:**
    - The heatmap shows a clear concentration of hospitals in the bin centered around latitude 51.085 to 51.671 and longitude -1.485 to -0.681. This bin has a significantly higher number of hospitals (202) compared to all other bins, indicated by the darkest shade of blue. This region likely corresponds to a major metropolitan area, possibly London or its immediate surroundings.
    - Several other bins, particularly those in the middle latitudes and longitudes, also show a moderate number of hospitals, suggesting a broader distribution of healthcare facilities across certain regions.
    - The bins at the extreme edges of the latitude and longitude ranges generally have very low or zero hospital counts. This implies that these areas are sparsely populated with hospitals, which could indicate limited healthcare access.
    - The heat map shows a uneven distrubution of hospitals across the given area.
    - The heat map also shows that there are some latitude and longitude bins that contain zero hospitals.

In summary, the heatmap reveals the uneven distribution of hospitals, with a clear concentration in a specific metropolitan region and sparser coverage in other areas. This information can be valuable for understanding healthcare access patterns and informing resource allocation decisions.

Strategic Recommendations Based on Hospital Data Analysis

1. Addressing Geographic Disparities in Healthcare Access:

- **Recommendation:** Prioritize resource allocation and development of new healthcare facilities in underserved areas identified by the geographic distribution analysis.
- **Rationale:** The heatmap clearly demonstrates an uneven distribution of hospitals, with significant concentrations in specific regions (e.g., likely London) and sparse coverage in others. This disparity may lead to unequal access to healthcare services, impacting

patient outcomes and overall public health. Strategic investment in underserved areas can help mitigate these inequalities.

2. Enhancing Data Completeness and Accuracy:

- **Recommendation:** Implement data governance policies and procedures to improve the completeness and accuracy of hospital contact information, particularly email addresses.
- **Rationale:** The analysis revealed a significant lack of email addresses for hospitals. This hinders efficient communication and may limit the use of digital health initiatives. Improving data completeness ensures better communication channels and facilitates more effective healthcare operations.

3. Strategic Planning for Hospital Services:

- **Recommendation:** Tailor healthcare service planning to the predominant hospital types and subtypes identified in the dataset.
- **Rationale:** The analysis indicates that "Hospital" is the most frequent OrganisationType and SubType. This suggests a focus on general hospital services. However, the presence of "Mental Health Hospitals" as the next frequent SubType highlights the need for specialized services. Planning should consider this distribution to ensure adequate provision of both general and specialized care.

4. Considering Sector Balance in Healthcare Provision:

- **Recommendation:** Policymakers should consider the balance between NHS Sector and Independent Sector hospitals when making decisions about healthcare provision and funding.
- **Rationale:** The analysis shows that the NHS Sector has a higher number of hospitals, but the Independent Sector also plays a significant role. Understanding the dynamics between these sectors is crucial for ensuring a comprehensive and sustainable healthcare system.

5. Monitoring the Influence of Parent Organizations:

- **Recommendation:** Healthcare regulators and policymakers should monitor the influence of large parent organizations on hospital distribution and service provision.

- **Rationale:** The analysis identified several parent organizations with a significant number of hospitals, indicating a degree of consolidation in the healthcare market. This concentration of ownership could have implications for competition, service diversity, and patient choice.

Conclusion

This analysis provides a comprehensive overview of the hospital data, highlighting key trends and patterns in hospital distribution, characteristics, and organizational structures. The findings from this report can be used to inform strategic decision-making in healthcare planning, resource allocation, and policy development. The use of Python, Excel, and Power BI facilitated efficient data processing, analysis, and visualization, enabling a clear and insightful presentation of the results.