In [1]:
```
# Standard Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Scikit-Learn Libraries
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
```

In [3]:
```
# Suppress warnings
import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
# 1. ETL Process - Load Datasets
# Load datasets
#staffing_df = pd.read_csv("PBJ_Daily_Nurse_Staffing_Q2_2024.csv")
staffing_df = pd.read_csv("PBJ_Daily_Nurse_Staffing_Q2_2024.csv", encoding='latin1'

staffing_df.head()
```

Out[4]:

| | PROVNUM | PROVNAME | CITY | STATE | COUNTY_NAME | COUNTY_FIPS | CY_Qtr |
|---|---|---|---|---|---|---|---|
| **0** | 15009 | BURNS NURSING HOME, INC. | RUSSELLVILLE | AL | Franklin | 59 | 2024Q2 |
| **1** | 15009 | BURNS NURSING HOME, INC. | RUSSELLVILLE | AL | Franklin | 59 | 2024Q2 |
| **2** | 15009 | BURNS NURSING HOME, INC. | RUSSELLVILLE | AL | Franklin | 59 | 2024Q2 |
| **3** | 15009 | BURNS NURSING HOME, INC. | RUSSELLVILLE | AL | Franklin | 59 | 2024Q2 |
| **4** | 15009 | BURNS NURSING HOME, INC. | RUSSELLVILLE | AL | Franklin | 59 | 2024Q2 |

5 rows × 33 columns

In [5]:
```
# 1. ETL Process - Load Datasets
# Load datasets
```

```
performance_df = pd.read_csv("FY_2025_SNF_VBP_Facility_Performance.csv")
performance_df.head()
```

Out[5]:

| | SNF VBP Program Ranking | Footnote -- SNF VBP Program Ranking | CMS Certification Number (CCN) | Provider Name | Provider Address | City/Town | State | ZIP Code | St R |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7099 | NaN | 10007 | MIZELL MEMORIAL HOSPITAL | 702 N MAIN ST | OPP | AL | 36467 | |
| **1** | 520 | NaN | 10044 | MARION REGIONAL MEDICAL CENTER | 1256 MILITARY STREET SOUTH | HAMILTON | AL | 35570 | |
| **2** | 1886 | NaN | 10045 | FAYETTE MEDICAL CENTER | 1653 TEMPLE AVENUE NORTH | FAYETTE | AL | 35555 | |
| **3** | 5795 | NaN | 10049 | MEDICAL CENTER ENTERPRISE | 400 N EDWARDS STREET | ENTERPRISE | AL | 36330 | |
| **4** | 5711 | NaN | 10058 | BIBB MEDICAL CENTER | 208 PIERSON AVE | CENTREVILLE | AL | 35042 | |

In [6]:
```
# 1. ETL Process - Load Datasets
# Load datasets
provider_info_df = pd.read_csv("NH_ProviderInfo_Nov2024.csv")
provider_info_df.head()
```

Out[6]:

| | CMS Certification Number (CCN) | Provider Name | Provider Address | City/Town | State | ZIP Code | Telephone Number | Pr |
|---|---|---|---|---|---|---|---|---|
| **0** | 015009 | BURNS NURSING HOME, INC. | 701 MONROE STREET NW | RUSSELLVILLE | AL | 35653 | 2563324110 | |
| **1** | 015010 | COOSA VALLEY HEALTHCARE CENTER | 260 WEST WALNUT STREET | SYLACAUGA | AL | 35150 | 2562495604 | |
| **2** | 015012 | HIGHLANDS HEALTH AND REHAB | 380 WOODS COVE ROAD | SCOTTSBORO | AL | 35768 | 2562183708 | |
| **3** | 015014 | EASTVIEW REHABILITATION & HEALTHCARE CENTER | 7755 FOURTH AVENUE SOUTH | BIRMINGHAM | AL | 35206 | 2058330146 | |
| **4** | 015015 | PLANTATION MANOR NURSING HOME | 6450 OLD TUSCALOOSA HIGHWAY | MC CALLA | AL | 35111 | 2054776161 | |

5 rows × 103 columns

In [7]:
```
# ETL Process - Load Datasets
import pandas as pd
import numpy as np

# Load datasets
staffing_df = pd.read_csv("PBJ_Daily_Nurse_Staffing_Q2_2024.csv", encoding='latin1'
performance_df = pd.read_csv("FY_2025_SNF_VBP_Facility_Performance.csv")
provider_info_df = pd.read_csv("NH_ProviderInfo_Nov2024.csv")

# Function to clean a dataframe
def clean_dataframe(df):
    # 1. Standardize column names
    df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")

    # 2. Handle missing values
    missing_values = df.isnull().sum()
    missing_threshold = 0.5  # Drop columns where more than 50% values are missing
    df = df.dropna(thresh=int(missing_threshold * len(df)), axis=1)

    # Fill missing numerical values with median
    num_cols = df.select_dtypes(include=['number']).columns
    df[num_cols] = df[num_cols].fillna(df[num_cols].median())

    # Fill missing categorical values with mode
```

```
    cat_cols = df.select_dtypes(include=['object']).columns
    for col in cat_cols:
        df[col].fillna(df[col].mode()[0], inplace=True)

    # 3. Remove duplicates
    df.drop_duplicates(inplace=True)

    # 4. Convert data types
    for col in num_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    for col in cat_cols:
        df[col] = df[col].astype(str).str.strip()

    # 5. Handle outliers (replace extreme values with 99th percentile)
    for col in num_cols:
        q99 = df[col].quantile(0.99)
        df[col] = np.where(df[col] > q99, q99, df[col])

    # 6. Remove whitespace & special characters from string fields
    df[cat_cols] = df[cat_cols].apply(lambda x: x.str.replace(r'[^a-zA-Z0-9 ]', '',

    return df

# Apply cleaning to all dataframes
staffing_df = clean_dataframe(staffing_df)
performance_df = clean_dataframe(performance_df)
provider_info_df = clean_dataframe(provider_info_df)

# Verify cleaning
print("Staffing Data Overview:\n", staffing_df.info())
print("Performance Data Overview:\n", performance_df.info())
print("Provider Info Data Overview:\n", provider_info_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1325324 entries, 0 to 1325323
Data columns (total 33 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   provnum           1325324 non-null  object
 1   provname          1325324 non-null  object
 2   city              1325324 non-null  object
 3   state             1325324 non-null  object
 4   county_name       1325324 non-null  object
 5   county_fips       1325324 non-null  float64
 6   cy_qtr            1325324 non-null  object
 7   workdate          1325324 non-null  float64
 8   mdscensus         1325324 non-null  float64
 9   hrs_rndon         1325324 non-null  float64
 10  hrs_rndon_emp     1325324 non-null  float64
 11  hrs_rndon_ctr     1325324 non-null  float64
 12  hrs_rnadmin       1325324 non-null  float64
 13  hrs_rnadmin_emp   1325324 non-null  float64
 14  hrs_rnadmin_ctr   1325324 non-null  float64
 15  hrs_rn            1325324 non-null  float64
 16  hrs_rn_emp        1325324 non-null  float64
 17  hrs_rn_ctr        1325324 non-null  float64
 18  hrs_lpnadmin      1325324 non-null  float64
 19  hrs_lpnadmin_emp  1325324 non-null  float64
 20  hrs_lpnadmin_ctr  1325324 non-null  float64
 21  hrs_lpn           1325324 non-null  float64
 22  hrs_lpn_emp       1325324 non-null  float64
 23  hrs_lpn_ctr       1325324 non-null  float64
 24  hrs_cna           1325324 non-null  float64
 25  hrs_cna_emp       1325324 non-null  float64
 26  hrs_cna_ctr       1325324 non-null  float64
 27  hrs_natrn         1325324 non-null  float64
 28  hrs_natrn_emp     1325324 non-null  float64
 29  hrs_natrn_ctr     1325324 non-null  float64
 30  hrs_medaide       1325324 non-null  float64
 31  hrs_medaide_emp   1325324 non-null  float64
 32  hrs_medaide_ctr   1325324 non-null  float64
dtypes: float64(27), object(6)
memory usage: 333.7+ MB
Staffing Data Overview:
 None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10533 entries, 0 to 10532
Data columns (total 13 columns):
 #   Column                                                    Non-Null Count
Dtype
---  ------                                                    --------------
-----
 0   snf_vbp_program_ranking                                   10533 non-null
float64
 1   cms_certification_number_(ccn)                            10533 non-null
float64
 2   provider_name                                             10533 non-null
object
 3   provider_address                                          10533 non-null
```

```
object
 4   city/town                                                    10533 non-null
object
 5   state                                                        10533 non-null
object
 6   zip_code                                                     10533 non-null
float64
 7   baseline_period:_fy_2019_risk-standardized_readmission_rate  10533 non-null
object
 8   performance_period:_fy_2023_risk-standardized_readmission_rate  10533 non-null
float64
 9   achievement_score                                            10533 non-null
float64
 10  improvement_score                                            10533 non-null
object
 11  performance_score                                            10533 non-null
float64
 12  incentive_payment_multiplier                                 10533 non-null
float64
dtypes: float64(7), object(6)
memory usage: 1.0+ MB
Performance Data Overview:
 None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14807 entries, 0 to 14806
Data columns (total 88 columns):
 #   Column                                                       Non-Null
Count  Dtype
---  ------                                                       --------
------  -----
 0   cms_certification_number_(ccn)                               14807 no
n-null  object
 1   provider_name                                                14807 no
n-null  object
 2   provider_address                                             14807 no
n-null  object
 3   city/town                                                    14807 no
n-null  object
 4   state                                                        14807 no
n-null  object
 5   zip_code                                                     14807 no
n-null  float64
 6   telephone_number                                             14807 no
n-null  float64
 7   provider_ssa_county_code                                     14807 no
n-null  float64
 8   county/parish                                                14807 no
n-null  object
 9   ownership_type                                               14807 no
n-null  object
 10  number_of_certified_beds                                     14807 no
n-null  float64
 11  average_number_of_residents_per_day                          14807 no
n-null  float64
 12  provider_type                                                14807 no
n-null  object
```

```
 13  provider_resides_in_hospital                                        14807 no
n-null  object
 14  legal_business_name                                                 14807 no
n-null  object
 15  date_first_approved_to_provide_medicare_and_medicaid_services       14807 no
n-null  object
 16  affiliated_entity_name                                              14807 no
n-null  object
 17  affiliated_entity_id                                                14807 no
n-null  float64
 18  continuing_care_retirement_community                                14807 no
n-null  object
 19  abuse_icon                                                          14807 no
n-null  object
 20  most_recent_health_inspection_more_than_2_years_ago                 14807 no
n-null  object
 21  provider_changed_ownership_in_last_12_months                        14807 no
n-null  object
 22  with_a_resident_and_family_council                                  14807 no
n-null  object
 23  automatic_sprinkler_systems_in_all_required_areas                   14807 no
n-null  object
 24  overall_rating                                                      14807 no
n-null  float64
 25  health_inspection_rating                                            14807 no
n-null  float64
 26  qm_rating                                                           14807 no
n-null  float64
 27  long-stay_qm_rating                                                 14807 no
n-null  float64
 28  short-stay_qm_rating                                                14807 no
n-null  float64
 29  staffing_rating                                                     14807 no
n-null  float64
 30  reported_nurse_aide_staffing_hours_per_resident_per_day             14807 no
n-null  float64
 31  reported_lpn_staffing_hours_per_resident_per_day                    14807 no
n-null  float64
 32  reported_rn_staffing_hours_per_resident_per_day                     14807 no
n-null  float64
 33  reported_licensed_staffing_hours_per_resident_per_day               14807 no
n-null  float64
 34  reported_total_nurse_staffing_hours_per_resident_per_day            14807 no
n-null  float64
 35  total_number_of_nurse_staff_hours_per_resident_per_day_on_the_weekend  14807 no
n-null  float64
 36  registered_nurse_hours_per_resident_per_day_on_the_weekend          14807 no
n-null  float64
 37  reported_physical_therapist_staffing_hours_per_resident_per_day     14807 no
n-null  float64
 38  total_nursing_staff_turnover                                        14807 no
n-null  float64
 39  registered_nurse_turnover                                           14807 no
n-null  float64
 40  number_of_administrators_who_have_left_the_nursing_home             14807 no
n-null  float64
```

```
 41  nursing_case-mix_index                                                14807 no
n-null  float64
 42  nursing_case-mix_index_ratio                                          14807 no
n-null  float64
 43  case-mix_nurse_aide_staffing_hours_per_resident_per_day               14807 no
n-null  float64
 44  case-mix_lpn_staffing_hours_per_resident_per_day                      14807 no
n-null  float64
 45  case-mix_rn_staffing_hours_per_resident_per_day                       14807 no
n-null  float64
 46  case-mix_total_nurse_staffing_hours_per_resident_per_day              14807 no
n-null  float64
 47  case-mix_weekend_total_nurse_staffing_hours_per_resident_per_day      14807 no
n-null  float64
 48  adjusted_nurse_aide_staffing_hours_per_resident_per_day               14807 no
n-null  float64
 49  adjusted_lpn_staffing_hours_per_resident_per_day                      14807 no
n-null  float64
 50  adjusted_rn_staffing_hours_per_resident_per_day                       14807 no
n-null  float64
 51  adjusted_total_nurse_staffing_hours_per_resident_per_day              14807 no
n-null  float64
 52  adjusted_weekend_total_nurse_staffing_hours_per_resident_per_day      14807 no
n-null  float64
 53  rating_cycle_1_standard_survey_health_date                            14807 no
n-null  object
 54  rating_cycle_1_total_number_of_health_deficiencies                    14807 no
n-null  float64
 55  rating_cycle_1_number_of_standard_health_deficiencies                 14807 no
n-null  float64
 56  rating_cycle_1_number_of_complaint_health_deficiencies                14807 no
n-null  float64
 57  rating_cycle_1_health_deficiency_score                                14807 no
n-null  float64
 58  rating_cycle_1_number_of_health_revisits                              14807 no
n-null  float64
 59  rating_cycle_1_health_revisit_score                                   14807 no
n-null  float64
 60  rating_cycle_1_total_health_score                                     14807 no
n-null  float64
 61  rating_cycle_2_standard_health_survey_date                            14807 no
n-null  object
 62  rating_cycle_2_total_number_of_health_deficiencies                    14807 no
n-null  float64
 63  rating_cycle_2_number_of_standard_health_deficiencies                 14807 no
n-null  float64
 64  rating_cycle_2_number_of_complaint_health_deficiencies                14807 no
n-null  float64
 65  rating_cycle_2_health_deficiency_score                                14807 no
n-null  float64
 66  rating_cycle_2_number_of_health_revisits                              14807 no
n-null  float64
 67  rating_cycle_2_health_revisit_score                                   14807 no
n-null  float64
 68  rating_cycle_2_total_health_score                                     14807 no
n-null  float64
```

```
 69  rating_cycle_3_standard_health_survey_date                          14807 no
n-null  object
 70  rating_cycle_3_total_number_of_health_deficiencies                  14807 no
n-null  float64
 71  rating_cycle_3_number_of_standard_health_deficiencies               14807 no
n-null  float64
 72  rating_cycle_3_number_of_complaint_health_deficiencies              14807 no
n-null  float64
 73  rating_cycle_3_health_deficiency_score                              14807 no
n-null  float64
 74  rating_cycle_3_number_of_health_revisits                            14807 no
n-null  float64
 75  rating_cycle_3_health_revisit_score                                 14807 no
n-null  float64
 76  rating_cycle_3_total_health_score                                   14807 no
n-null  float64
 77  total_weighted_health_survey_score                                  14807 no
n-null  float64
 78  number_of_facility_reported_incidents                               14807 no
n-null  float64
 79  number_of_substantiated_complaints                                  14807 no
n-null  float64
 80  number_of_fines                                                     14807 no
n-null  float64
 81  total_amount_of_fines_in_dollars                                    14807 no
n-null  float64
 82  number_of_payment_denials                                           14807 no
n-null  float64
 83  total_number_of_penalties                                           14807 no
n-null  float64
 84  location                                                            14807 no
n-null  object
 85  latitude                                                            14807 no
n-null  float64
 86  longitude                                                           14807 no
n-null  float64
 87  processing_date                                                     14807 no
n-null  object
dtypes: float64(65), object(23)
memory usage: 9.9+ MB
Provider Info Data Overview:
 None
```

In [ ]:

In [8]:
```
# Adjusted column names based on standardization
staffing_cols = ["provnum", "state", "workdate", "mdscensus", "hrs_rn", "hrs_lpn",
performance_cols = ["cms_certification_number_(ccn)", "performance_score", "incenti
                    "baseline_period:_fy_2019_risk-standardized_readmission_rate",
                    "performance_period:_fy_2023_risk-standardized_readmission_rate
provider_cols = ["cms_certification_number_(ccn)", "number_of_certified_beds", "own

# Select relevant columns
staffing_df = staffing_df[staffing_cols]
performance_df = performance_df[performance_cols]
provider_info_df = provider_info_df[provider_cols]
```

```
# Verify selection
print("Staffing Data Sample:\n", staffing_df.head())
print("Performance Data Sample:\n", performance_df.head())
print("Provider Info Data Sample:\n", provider_info_df.head())

# Convert 'provnum' to string to match 'cms_certification_number_(ccn)' for merging
staffing_df['provnum'] = staffing_df['provnum'].astype(str)
performance_df['cms_certification_number_(ccn)'] = performance_df['cms_certificatio
provider_info_df['cms_certification_number_(ccn)'] = provider_info_df['cms_certific

# First merge: Merge staffing_df and performance_df on 'provnum' and 'cms_certifica
merged_df = pd.merge(staffing_df, performance_df, how="outer", left_on="provnum", r

# Second merge: Merge the result with provider_info_df on 'provnum' and 'cms_certif
merged_df = pd.merge(merged_df, provider_info_df, how="outer", left_on="provnum", r

# Verify the merged dataset
print("Merged Data Sample:\n", merged_df.head())
```

```
Staffing Data Sample:
   provnum state    workdate  mdscensus  hrs_rn  hrs_lpn  hrs_cna
0   15009    AL  20240401.0       51.0   55.70    25.50   160.08
1   15009    AL  20240402.0       52.0   63.28    15.22   135.95
2   15009    AL  20240403.0       53.0   76.29     5.46   150.31
3   15009    AL  20240404.0       52.0   54.13    20.18   133.01
4   15009    AL  20240405.0       52.0   53.63    27.85   137.92
Performance Data Sample:
    cms_certification_number_(ccn)  performance_score  \
0                          10007.0            0.00000
1                          10044.0           92.23297
2                          10045.0           62.63053
3                          10049.0           18.76358
4                          10058.0           19.71773

   incentive_payment_multiplier  \
0                      0.980256
1                      1.017769
2                      1.009874
3                      0.981615
4                      0.981769

  baseline_period:_fy_2019_risk-standardized_readmission_rate  \
0                                             020151
1                                             017141
2                                             019029
3
4                                             018396

   performance_period:_fy_2023_risk-standardized_readmission_rate
0                                           0.258692
1                                           0.172040
2                                           0.184450
3                                           0.202840
4                                           0.202440
Provider Info Data Sample:
   cms_certification_number_(ccn)  number_of_certified_beds  \
0                          015009                      57.0
1                          015010                      85.0
2                          015012                      50.0
3                          015014                      92.0
4                          015015                     103.0

           ownership_type  qm_rating
0  For profit  Corporation        4.0
1  For profit  Corporation        3.0
2       Government  County        2.0
3   For profit  Individual        2.0
4   For profit  Individual        2.0
Merged Data Sample:
   provnum state    workdate  mdscensus  hrs_rn  hrs_lpn  hrs_cna  \
0   15009    AL  20240401.0       51.0   55.70    25.50   160.08
1   15009    AL  20240402.0       52.0   63.28    15.22   135.95
2   15009    AL  20240403.0       53.0   76.29     5.46   150.31
3   15009    AL  20240404.0       52.0   54.13    20.18   133.01
4   15009    AL  20240405.0       52.0   53.63    27.85   137.92
```

```
  cms_certification_number_(ccn)_x  performance_score  \
0                              NaN                NaN
1                              NaN                NaN
2                              NaN                NaN
3                              NaN                NaN
4                              NaN                NaN

   incentive_payment_multiplier  \
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN

  baseline_period:_fy_2019_risk-standardized_readmission_rate  \
0                                                  NaN
1                                                  NaN
2                                                  NaN
3                                                  NaN
4                                                  NaN

   performance_period:_fy_2023_risk-standardized_readmission_rate  \
0                                                  NaN
1                                                  NaN
2                                                  NaN
3                                                  NaN
4                                                  NaN

  cms_certification_number_(ccn)_y  number_of_certified_beds ownership_type  \
0                              NaN                       NaN            NaN
1                              NaN                       NaN            NaN
2                              NaN                       NaN            NaN
3                              NaN                       NaN            NaN
4                              NaN                       NaN            NaN

   qm_rating
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
```

In [ ]:

In [9]:
```
# 1. Convert empty strings to NaN in the problematic columns
merged_df['baseline_period:_fy_2019_risk-standardized_readmission_rate'] = \
    pd.to_numeric(merged_df['baseline_period:_fy_2019_risk-standardized_readmission

merged_df['performance_period:_fy_2023_risk-standardized_readmission_rate'] = \
    pd.to_numeric(merged_df['performance_period:_fy_2023_risk-standardized_readmiss

# 2. Handle missing values by filling them with the median of the column
merged_df['baseline_period:_fy_2019_risk-standardized_readmission_rate'].fillna(
    merged_df['baseline_period:_fy_2019_risk-standardized_readmission_rate'].median
```

```
merged_df['performance_period:_fy_2023_risk-standardized_readmission_rate'].fillna(
    merged_df['performance_period:_fy_2023_risk-standardized_readmission_rate'].med

# 3. After filling missing values, continue with other operations as needed
print(merged_df.head())
```

```
  provnum state    workdate  mdscensus  hrs_rn  hrs_lpn  hrs_cna  \
0   15009    AL  20240401.0       51.0   55.70    25.50   160.08
1   15009    AL  20240402.0       52.0   63.28    15.22   135.95
2   15009    AL  20240403.0       53.0   76.29     5.46   150.31
3   15009    AL  20240404.0       52.0   54.13    20.18   133.01
4   15009    AL  20240405.0       52.0   53.63    27.85   137.92

  cms_certification_number_(ccn)_x  performance_score  \
0                              NaN                NaN
1                              NaN                NaN
2                              NaN                NaN
3                              NaN                NaN
4                              NaN                NaN

   incentive_payment_multiplier  \
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN

   baseline_period:_fy_2019_risk-standardized_readmission_rate  \
0                                            19812.0
1                                            19812.0
2                                            19812.0
3                                            19812.0
4                                            19812.0

   performance_period:_fy_2023_risk-standardized_readmission_rate  \
0                                            0.20236
1                                            0.20236
2                                            0.20236
3                                            0.20236
4                                            0.20236

  cms_certification_number_(ccn)_y  number_of_certified_beds ownership_type  \
0                              NaN                       NaN            NaN
1                              NaN                       NaN            NaN
2                              NaN                       NaN            NaN
3                              NaN                       NaN            NaN
4                              NaN                       NaN            NaN

   qm_rating
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
```

In [ ]:

In [10]:
```
# Drop the redundant columns resulting from the merge
merged_df.drop(columns=['cms_certification_number_(ccn)_x', 'cms_certification_numb

# Optionally, rename columns to avoid confusion
merged_df.rename(columns={
    'performance_score': 'performance_score',
    'incentive_payment_multiplier': 'incentive_payment_multiplier',
    'baseline_period:_fy_2019_risk-standardized_readmission_rate': 'baseline_readmi
    'performance_period:_fy_2023_risk-standardized_readmission_rate': 'performance_
    'number_of_certified_beds': 'certified_beds',
    'ownership_type': 'ownership_type',
    'qm_rating': 'quality_rating'
}, inplace=True)

# Check the resulting dataframe
print(merged_df.head())
```

```
  provnum state    workdate  mdscensus  hrs_rn  hrs_lpn  hrs_cna  \
0   15009    AL  20240401.0       51.0   55.70    25.50   160.08
1   15009    AL  20240402.0       52.0   63.28    15.22   135.95
2   15009    AL  20240403.0       53.0   76.29     5.46   150.31
3   15009    AL  20240404.0       52.0   54.13    20.18   133.01
4   15009    AL  20240405.0       52.0   53.63    27.85   137.92

   performance_score  incentive_payment_multiplier  \
0                NaN                           NaN
1                NaN                           NaN
2                NaN                           NaN
3                NaN                           NaN
4                NaN                           NaN

   baseline_readmission_rate_2019  performance_readmission_rate_2023  \
0                         19812.0                            0.20236
1                         19812.0                            0.20236
2                         19812.0                            0.20236
3                         19812.0                            0.20236
4                         19812.0                            0.20236

   certified_beds ownership_type  quality_rating
0             NaN            NaN             NaN
1             NaN            NaN             NaN
2             NaN            NaN             NaN
3             NaN            NaN             NaN
4             NaN            NaN             NaN
```

In [ ]:
```
```

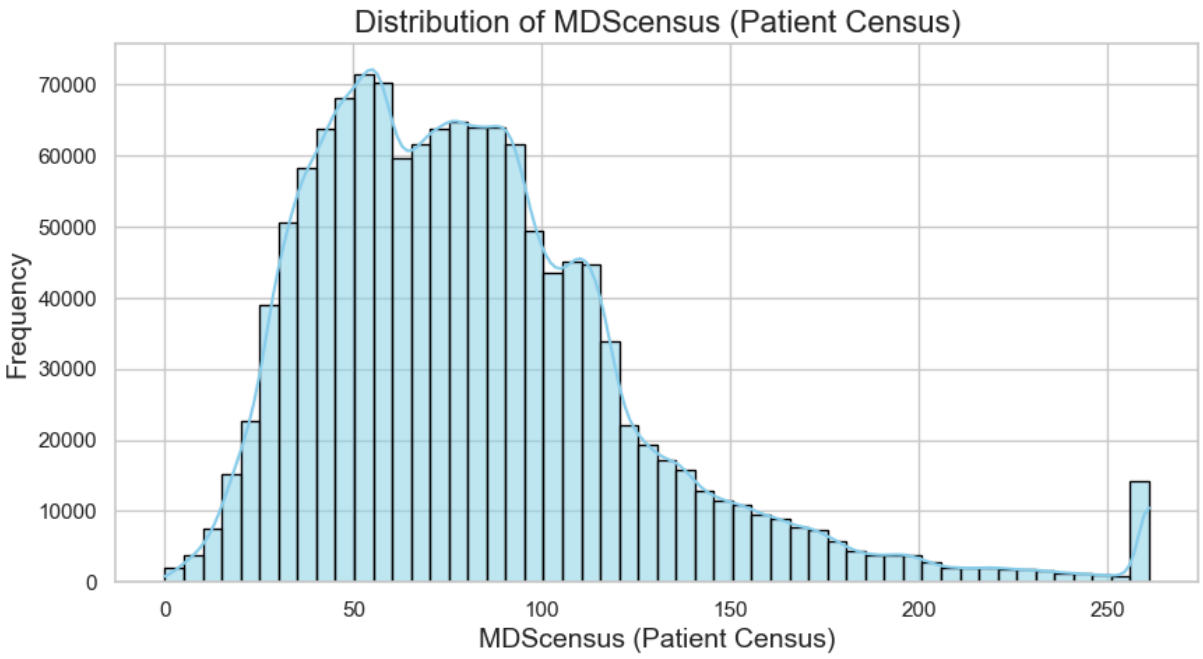
In [11]:
```
# Basic statistics
print("Basic Statistics:\n", merged_df.describe())
```

```
Basic Statistics:
           workdate     mdscensus        hrs_rn       hrs_lpn       hrs_cna  \
count  1.325324e+06  1.325324e+06  1.325324e+06  1.325324e+06  1.325324e+06
mean   2.024052e+07  8.264872e+01  3.403280e+01  6.553013e+01  1.718204e+02
std    8.167078e+01  4.472823e+01  2.995932e+01  4.521497e+01  1.040330e+02
min    2.024040e+07  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%    2.024042e+07  5.100000e+01  1.325000e+01  3.260000e+01  9.857000e+01
50%    2.024052e+07  7.600000e+01  2.575000e+01  5.681000e+01  1.506200e+02
75%    2.024061e+07  1.040000e+02  4.525000e+01  8.850000e+01  2.200000e+02
max    2.024063e+07  2.610000e+02  1.659754e+02  2.322500e+02  5.977500e+02

       performance_score  incentive_payment_multiplier  \
count       10533.000000                  10533.000000
mean           30.857544                      0.991145
std            30.656148                      0.013751
min             0.000000                      0.980256
25%             0.000000                      0.980256
50%            24.262210                      0.982715
75%            53.466910                      1.002450
max           100.000000                      1.018066

       baseline_readmission_rate_2019  performance_readmission_rate_2023  \
count                    1.337736e+06                       1.337736e+06
mean                     1.981293e+04                       2.023684e-01
std                      1.666939e+02                       1.801814e-03
min                      1.265200e+04                       1.167900e-01
25%                      1.981200e+04                       2.023600e-01
50%                      1.981200e+04                       2.023600e-01
75%                      1.981200e+04                       2.023600e-01
max                      3.014300e+04                       2.586924e-01

       certified_beds  quality_rating
count    1.177654e+06    1.177654e+06
mean     1.055947e+02    3.391388e+00
std      5.321631e+01    1.286011e+00
min      4.000000e+00    1.000000e+00
25%      6.500000e+01    2.000000e+00
50%      1.000000e+02    4.000000e+00
75%      1.260000e+02    4.000000e+00
max      3.050000e+02    5.000000e+00
```

In [ ]:

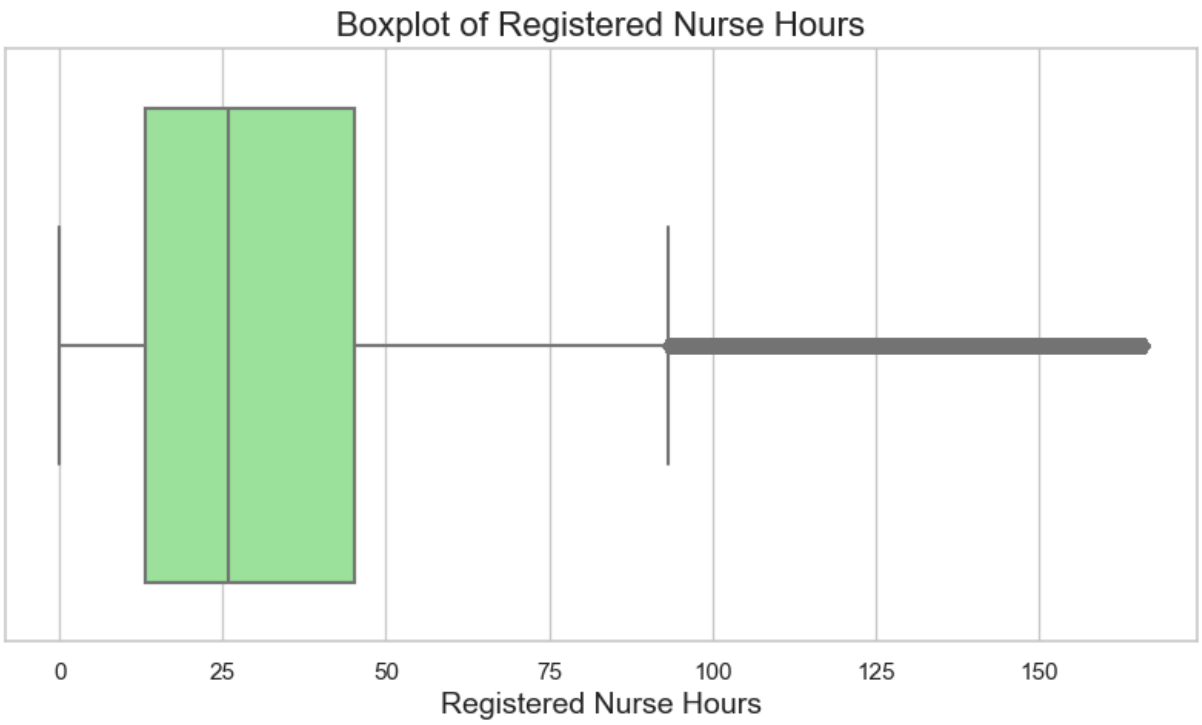
In [12]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Automatically calculate number of bins based on the data
num_bins = int(merged_df["mdscensus"].nunique() / 5)

# Set plot size and style
plt.figure(figsize=(10, 5))
sns.set(style="whitegrid")

# Plotting histogram with KDE for smooth distribution visualization
sns.histplot(merged_df["mdscensus"], bins=num_bins, kde=True, color="skyblue", edge
```

```
# Adding title and labels
plt.title("Distribution of MDScensus (Patient Census)", fontsize=16)
plt.xlabel("MDScensus (Patient Census)", fontsize=14)
plt.ylabel("Frequency", fontsize=14)

# Show plot
plt.show()
```

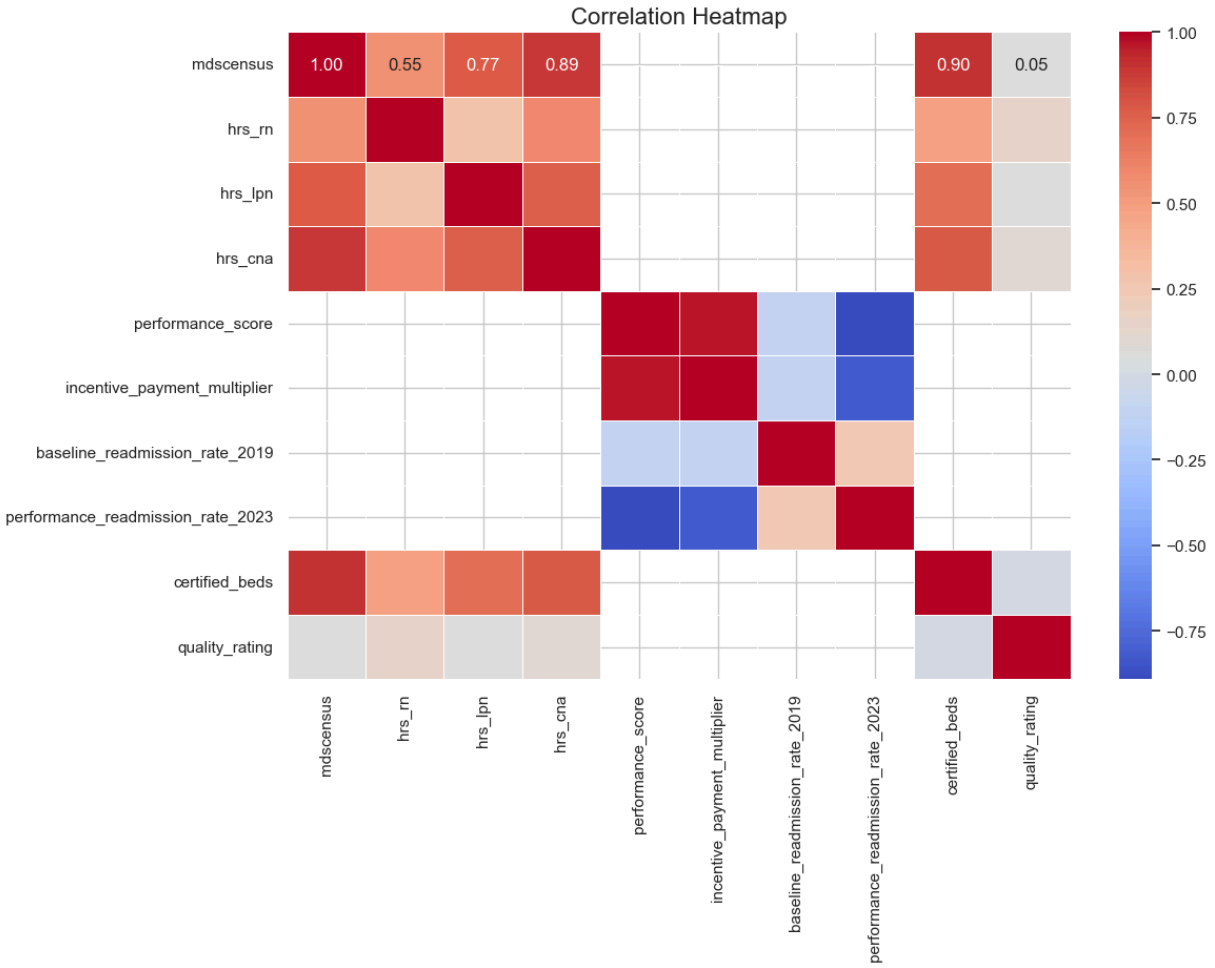



In [ ]:

In [13]:
```
# 2. Boxplot of hrs_rn (Registered Nurse Hours)
plt.figure(figsize=(10, 5))
sns.boxplot(x=merged_df["hrs_rn"], color="lightgreen")
plt.title("Boxplot of Registered Nurse Hours", fontsize=16)
plt.xlabel("Registered Nurse Hours", fontsize=14)
plt.show()
```

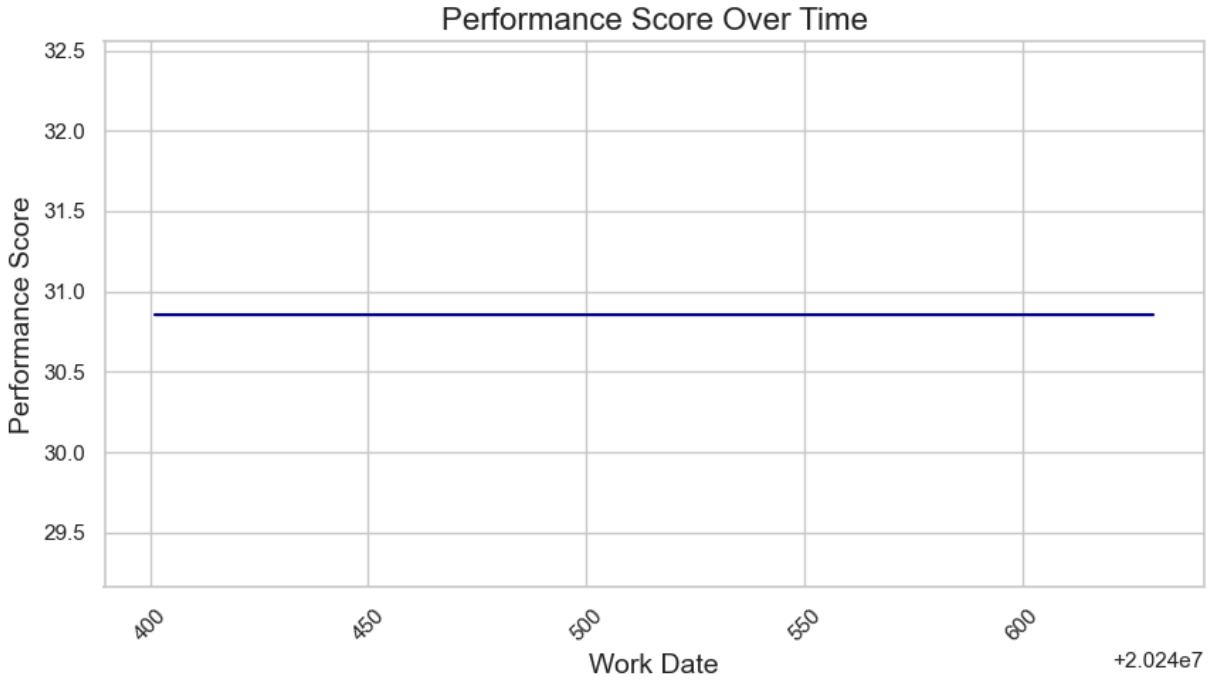



In [14]:
```
# 3. Correlation Heatmap between numerical variables
correlation_matrix = merged_df[["mdscensus", "hrs_rn", "hrs_lpn", "hrs_cna", "perfo
                                "incentive_payment_multiplier", "baseline_readmissi
                                "performance_readmission_rate_2023", "certified_bed
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=
plt.title("Correlation Heatmap", fontsize=16)
plt.show()
```

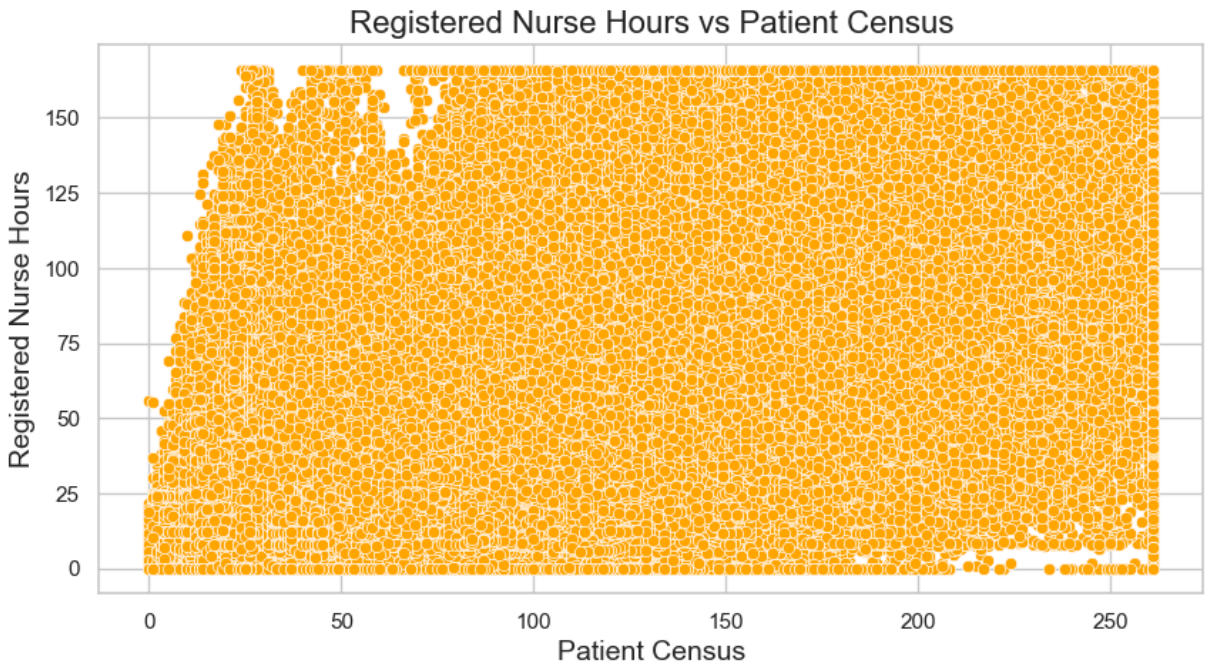



In [15]:
```
# Impute with mean (or median, mode depending on context)
merged_df['performance_score'].fillna(merged_df['performance_score'].mean(), inplac
```

In [16]:
```
# Line Plot of performance_score over time (workdate)
plt.figure(figsize=(10, 5))
sns.lineplot(x='workdate', y='performance_score', data=merged_df, color='darkblue')
plt.title("Performance Score Over Time", fontsize=16)
plt.xlabel("Work Date", fontsize=14)
plt.ylabel("Performance Score", fontsize=14)
plt.xticks(rotation=45)
plt.show()
```

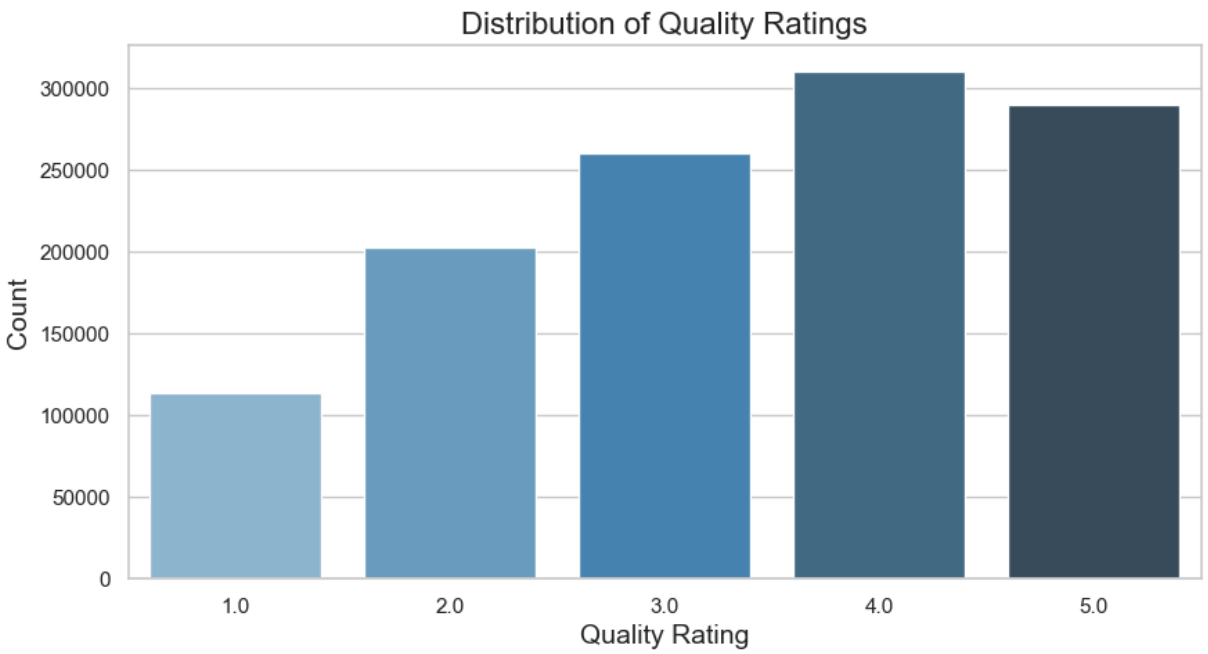



In [17]:
```
# 5. Scatter Plot of hrs_rn vs mdscensus (Registered Nurse Hours vs Patient Census)
plt.figure(figsize=(10, 5))
sns.scatterplot(x=merged_df["mdscensus"], y=merged_df["hrs_rn"], color="orange")
plt.title("Registered Nurse Hours vs Patient Census", fontsize=16)
plt.xlabel("Patient Census", fontsize=14)
plt.ylabel("Registered Nurse Hours", fontsize=14)
plt.show()
```




In [18]:
```
# 6. Bar Plot of quality_rating
plt.figure(figsize=(10, 5))
sns.countplot(x=merged_df["quality_rating"], palette="Blues_d")
plt.title("Distribution of Quality Ratings", fontsize=16)
plt.xlabel("Quality Rating", fontsize=14)
```

```
plt.ylabel("Count", fontsize=14)
plt.show()
```




In [ ]:

In [ ]:

In [ ]:

In [19]:
```
# Print the column names of merged_df
print(merged_df.columns)
```

```
Index(['provnum', 'state', 'workdate', 'mdscensus', 'hrs_rn', 'hrs_lpn',
       'hrs_cna', 'performance_score', 'incentive_payment_multiplier',
       'baseline_readmission_rate_2019', 'performance_readmission_rate_2023',
       'certified_beds', 'ownership_type', 'quality_rating'],
      dtype='object')
```

In [ ]:

In [ ]:

In [20]:
```
# 4. Feature Engineering
# Convert categorical feature 'Ownership Type' into numerical encoding
le = LabelEncoder()
merged_df["ownership_type"] = le.fit_transform(merged_df["ownership_type"])
```

In [21]:
```
# Create new features
merged_df["Revenue_per_Bed"] = merged_df["mdscensus"] / merged_df["certified_beds"]
merged_df["Revenue_per_Nurse_Hour"] = merged_df["mdscensus"] / (merged_df["hrs_rn"]
merged_df["Staffing_Efficiency_Ratio"] = merged_df["hrs_cna"] / merged_df["mdscensu
```

```
# Display the first few rows to verify the new features
print(merged_df.head())
```

```
  provnum state    workdate  mdscensus  hrs_rn  hrs_lpn  hrs_cna  \
0   15009    AL  20240401.0       51.0   55.70    25.50   160.08
1   15009    AL  20240402.0       52.0   63.28    15.22   135.95
2   15009    AL  20240403.0       53.0   76.29     5.46   150.31
3   15009    AL  20240404.0       52.0   54.13    20.18   133.01
4   15009    AL  20240405.0       52.0   53.63    27.85   137.92

   performance_score  incentive_payment_multiplier  \
0          30.857544                           NaN
1          30.857544                           NaN
2          30.857544                           NaN
3          30.857544                           NaN
4          30.857544                           NaN

   baseline_readmission_rate_2019  performance_readmission_rate_2023  \
0                         19812.0                            0.20236
1                         19812.0                            0.20236
2                         19812.0                            0.20236
3                         19812.0                            0.20236
4                         19812.0                            0.20236

   certified_beds  ownership_type  quality_rating  Revenue_per_Bed  \
0             NaN             13             NaN              NaN
1             NaN             13             NaN              NaN
2             NaN             13             NaN              NaN
3             NaN             13             NaN              NaN
4             NaN             13             NaN              NaN

   Revenue_per_Nurse_Hour  Staffing_Efficiency_Ratio
0                0.211373                   3.138824
1                0.242481                   2.614423
2                0.228389                   2.836038
3                0.250820                   2.557885
4                0.237010                   2.652308
```

In [22]:
```
# Fill missing values with median only for numeric columns
numeric_cols = merged_df.select_dtypes(include=['number']).columns
merged_df[numeric_cols] = merged_df[numeric_cols].fillna(merged_df[numeric_cols].me
```

In [23]:
```
# 5. Data Preprocessing
# Define features and target variable
X = merged_df.drop(columns=["mdscensus"])  # Predicting mdscensus (proxy for revenu
y = merged_df["mdscensus"]
```

In [24]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [25]:
```
import numpy as np

# Replace infinite values with NaN
X_train.replace([np.inf, -np.inf], np.nan, inplace=True)
X_test.replace([np.inf, -np.inf], np.nan, inplace=True)
```

```
# Select only numeric columns to calculate the mean and fill NaN values
numeric_cols = X_train.select_dtypes(include=[np.number]).columns

# Fill NaN values with the column mean (only for numeric columns)
X_train[numeric_cols] = X_train[numeric_cols].fillna(X_train[numeric_cols].mean())
X_test[numeric_cols] = X_test[numeric_cols].fillna(X_test[numeric_cols].mean())
```

In [26]:

```
# using few sample dataset
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Sample the data (1% sample size for demonstration)
X_train_sampled, _, y_train_sampled, _ = train_test_split(X_train, y_train, train_s
X_test_sampled, _, y_test_sampled, _ = train_test_split(X_test, y_test, train_size=

# Separate numeric and categorical columns
numeric_cols = X_train_sampled.select_dtypes(include=[np.number]).columns
categorical_cols = X_train_sampled.select_dtypes(exclude=[np.number]).columns

# Create transformers for numerical and categorical data
numeric_transformer = SimpleImputer(strategy='mean')
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Create a preprocessor that applies the right transformations to each type of feat
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

# Initialize models with reduced complexity
models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(n_estimators=20, random_state=42, n_jobs
    "Gradient Boosting": GradientBoostingRegressor(n_estimators=50, learning_rate=0
}

# Train and evaluate models
results = {}
for name, model in models.items():
    # Create a pipeline that applies pre-processing and then the model
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('model', model)])

    pipeline.fit(X_train_sampled, y_train_sampled)  # Train on sampled data
```

```
    y_pred = pipeline.predict(X_test_sampled)  # Predict on test data

    # Store performance metrics
    results[name] = {
        "R2 Score": r2_score(y_test_sampled, y_pred),
        "MAE": mean_absolute_error(y_test_sampled, y_pred),
        "MSE": mean_squared_error(y_test_sampled, y_pred)
    }

# Print results
for name, metrics in results.items():
    print(f"{name}: {metrics}")
```

```
Linear Regression: {'R2 Score': 0.9669847166576218, 'MAE': 4.057660078647335, 'MSE':
68.5030968019308}
Random Forest: {'R2 Score': 0.9961685048817334, 'MAE': 0.9252710280373833, 'MSE': 7.
949932710280374}
Gradient Boosting: {'R2 Score': 0.9802402050374126, 'MAE': 3.9250404541974535, 'MS
E': 40.999410275269426}
```

In [27]:
```
# Display results
for name, metrics in results.items():
    print(f"Model: {name}")
    print(f"R2 Score: {metrics['R2 Score']:.4f}")
    print(f"MAE: {metrics['MAE']:.2f}")
    print(f"MSE: {metrics['MSE']:.2f}\n")
```

```
Model: Linear Regression
R2 Score: 0.9670
MAE: 4.06
MSE: 68.50

Model: Random Forest
R2 Score: 0.9962
MAE: 0.93
MSE: 7.95

Model: Gradient Boosting
R2 Score: 0.9802
MAE: 3.93
MSE: 41.00
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```
# 8. Model Performance Enhancement - Hyperparameter Tuning (Gradient Boosting)
# ------------------------
param_grid = {
    "n_estimators": [100, 200, 300],
    "learning_rate": [0.01, 0.05, 0.1],
    "max_depth": [3, 5, 7]
}
```

```
gb_model = GradientBoostingRegressor(random_state=42)
grid_search = GridSearchCV(gb_model, param_grid, cv=3, scoring="r2", n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best parameters
print("Best parameters for Gradient Boosting:", grid_search.best_params_)
```

In [ ]:
```
# Retrain with best parameters
best_gb_model = grid_search.best_estimator_
y_pred_best = best_gb_model.predict(X_test)

# Evaluate performance
print(f"Optimized Gradient Boosting R2 Score: {r2_score(y_test, y_pred_best):.4f}")
print(f"Optimized Gradient Boosting MAE: {mean_absolute_error(y_test, y_pred_best):
print(f"Optimized Gradient Boosting MSE: {mean_squared_error(y_test, y_pred_best):.
```

In [ ]:
```

```

In [ ]:
```

```