In [2]:
```
import pandas as pd

# Load the datasets
provider_info_df = pd.read_csv('NH_ProviderInfo_Aug2024 - Copy.csv',low_memory=Fals
staffing_data_df = pd.read_csv('PBJ_Daily_Nurse_Staffing_Q1_2024 - Copy.csv',low_me
```

In [ ]:
```
```

In [3]:
```
provider_info_df.dtypes
```

Out[3]:
```
CCN                                                                object
Provider Name                                                      object
City/Town                                                          object
State                                                              object
County/Parish                                                      object
Ownership Type                                                     object
Number of Certified Beds                                            int64
Provider Type                                                      object
Overall Rating                                                      int64
Staffing Rating                                                     int64
Reported Nurse Aide Staffing Hours per Resident per Day           float64
Reported LPN Staffing Hours per Resident per Day                  float64
Reported RN Staffing Hours per Resident per Day                   float64
Reported Licensed Staffing Hours per Resident per Day             float64
Reported Total Nurse Staffing Hours per Resident per Day          float64
Total number of nurse staff hours per resident per day on the weekend    float64
Registered Nurse hours per resident per day on the weekend        float64
Reported Physical Therapist Staffing Hours per Resident Per Day   float64
dtype: object
```

In [4]:
```
staffing_data_df.dtypes
```

Out[4]:
```
PROVNUM              object
WorkDate              int64
Hrs_RNDON           float64
Hrs_RNDON_emp       float64
Hrs_RNDON_ctr       float64
Hrs_RNadmin         float64
Hrs_RNadmin_emp     float64
Hrs_RNadmin_ctr     float64
Hrs_RN              float64
Hrs_RN_emp          float64
Hrs_RN_ctr          float64
Hrs_LPNadmin        float64
Hrs_LPNadmin_emp    float64
Hrs_LPNadmin_ctr    float64
Hrs_LPN             float64
Hrs_LPN_emp         float64
Hrs_LPN_ctr         float64
Hrs_CNA             float64
Hrs_CNA_emp         float64
Hrs_CNA_ctr         float64
Hrs_NAtrn           float64
Hrs_NAtrn_emp       float64
Hrs_NAtrn_ctr       float64
Hrs_MedAide         float64
Hrs_MedAide_emp     float64
Hrs_MedAide_ctr     float64
dtype: object
```

In [5]:
```
staffing_summary = staffing_data_df.groupby('PROVNUM').agg(
    total_RN_emp=('Hrs_RN_emp', 'sum'),
    total_RN_ctr=('Hrs_RN_ctr', 'sum'),
    total_LPN_emp=('Hrs_LPN_emp', 'sum'),
    total_LPN_ctr=('Hrs_LPN_ctr', 'sum'),
    total_CNA_emp=('Hrs_CNA_emp', 'sum'),
    total_CNA_ctr=('Hrs_CNA_ctr', 'sum'),
).reset_index()
print(staffing_summary)
```

```
      PROVNUM  total_RN_emp  total_RN_ctr  total_LPN_emp  total_LPN_ctr  \
0      01A193       1131.01           0.0        3729.98            0.0
1      01A208       2331.00          28.5        3328.00            0.0
2      04A158       2599.11           0.0        7043.19            0.0
3      04A293       9078.50         937.0        5899.25        11267.0
4      05A024       2038.43           0.0       16089.00            0.0
...       ...           ...           ...            ...            ...
11518   95031       9935.25           0.0       11708.75            0.0
11519   95034       8387.51           0.0       15303.84            0.0
11520   95036      14178.00           0.0        8966.00            0.0
11521   95038       6031.25         180.0        2325.00           12.0
11522   95040       2951.70         979.0           0.00            0.0

       total_CNA_emp  total_CNA_ctr
0           10937.32           0.00
1            6536.25          66.98
2           28663.67           0.00
3           40097.75        4689.75
4           31446.08           0.00
...              ...            ...
11518       36870.00           0.00
11519       28773.50           0.00
11520       51221.50           0.00
11521       10903.25         120.00
11522        3653.94           0.00

[11523 rows x 7 columns]
```

In [6]:
```
staffing_data_df.dtypes
```

Out[6]:
```
PROVNUM              object
WorkDate              int64
Hrs_RNDON           float64
Hrs_RNDON_emp       float64
Hrs_RNDON_ctr       float64
Hrs_RNadmin         float64
Hrs_RNadmin_emp     float64
Hrs_RNadmin_ctr     float64
Hrs_RN              float64
Hrs_RN_emp          float64
Hrs_RN_ctr          float64
Hrs_LPNadmin        float64
Hrs_LPNadmin_emp    float64
Hrs_LPNadmin_ctr    float64
Hrs_LPN             float64
Hrs_LPN_emp         float64
Hrs_LPN_ctr         float64
Hrs_CNA             float64
Hrs_CNA_emp         float64
Hrs_CNA_ctr         float64
Hrs_NAtrn           float64
Hrs_NAtrn_emp       float64
Hrs_NAtrn_ctr       float64
Hrs_MedAide         float64
Hrs_MedAide_emp     float64
Hrs_MedAide_ctr     float64
dtype: object
```

In [7]:
```
# Sample from the provider_info_df DataFrame
provider_info_sample = provider_info_df.sample(n=2000, random_state=1)  # Sample si

# Sample from the staffing_data_df DataFrame
staffing_data_sample = staffing_data_df.sample(n=50000, random_state=1)  # Sample s
```

In [8]:
```
# Save the sampled data to new CSV files
provider_info_sample.to_csv('provider_info_sample.csv', index=False)
staffing_data_sample.to_csv('staffing_data_sample.csv', index=False)

print("Sampling completed and files saved.")
```

```
Sampling completed and files saved.
```

In [9]:
```
# Grouping by facility (PROVNUM) and calculating total hours for employees (emp) an
staffing_summary = staffing_data_sample.groupby('PROVNUM').agg({
    'Hrs_RNDON_emp': 'sum', 'Hrs_RNDON_ctr': 'sum',
    'Hrs_RNadmin_emp': 'sum', 'Hrs_RNadmin_ctr': 'sum',
    'Hrs_RN_emp': 'sum', 'Hrs_RN_ctr': 'sum',
    'Hrs_LPNadmin_emp': 'sum', 'Hrs_LPNadmin_ctr': 'sum',
    'Hrs_LPN_emp': 'sum', 'Hrs_LPN_ctr': 'sum',
    'Hrs_CNA_emp': 'sum', 'Hrs_CNA_ctr': 'sum',
    'Hrs_NAtrn_emp': 'sum', 'Hrs_NAtrn_ctr': 'sum',
    'Hrs_MedAide_emp': 'sum', 'Hrs_MedAide_ctr': 'sum'
})

# Summing the total employee and contractor hours across all roles (RN, LPN, CNA, e
staffing_summary['Total_Emp_Hours'] = staffing_summary.filter(like='_emp').sum(axis
```

```
staffing_summary['Total_Ctr_Hours'] = staffing_summary.filter(like='_ctr').sum(axis


# Displaying the first few rows of the summary
print(staffing_summary.head())
```

```
         Hrs_RNDON_emp  Hrs_RNDON_ctr  Hrs_RNadmin_emp  Hrs_RNadmin_ctr  \
PROVNUM
01A193            3.93            0.0              0.0              0.0
01A208            8.00            0.0              0.0              0.0
04A158           40.00            0.0              0.0              0.0
04A293           17.00            0.0            112.5              0.0
05A024            8.00            0.0              0.0              0.0

         Hrs_RN_emp  Hrs_RN_ctr  Hrs_LPNadmin_emp  Hrs_LPNadmin_ctr  \
PROVNUM
01A193        37.33         0.0              0.00               0.0
01A208        78.50         0.0             12.50               0.0
04A158       192.37         0.0              0.00               0.0
04A293       323.50        36.0             34.75               0.0
05A024        30.47         0.0             36.00               0.0

         Hrs_LPN_emp  Hrs_LPN_ctr  Hrs_CNA_emp  Hrs_CNA_ctr  Hrs_NAtrn_emp  \
PROVNUM
01A193        133.28         0.00       368.87          0.0           0.00
01A208        161.25         0.00       266.00          0.0           0.00
04A158        630.32         0.00      2489.75          0.0           0.00
04A293        285.50       492.25      1908.00        170.5          54.75
05A024        188.52         0.00       333.76          0.0          38.64

         Hrs_NAtrn_ctr  Hrs_MedAide_emp  Hrs_MedAide_ctr  Total_Emp_Hours  \
PROVNUM
01A193             0.0              0.0              0.0           543.41
01A208             0.0              0.0              0.0           526.25
04A158             0.0              0.0              0.0          3352.44
04A293             0.0              0.0              0.0          2736.00
05A024             0.0              0.0              0.0           635.39

         Total_Ctr_Hours
PROVNUM
01A193              0.00
01A208              0.00
04A158              0.00
04A293            698.75
05A024              0.00
```

In [10]:
```
staffing_summary_reset = staffing_summary.reset_index()
```

In [11]:
```
merged_df = pd.merge(provider_info_df ,staffing_data_sample, left_on='CCN', right_o
```

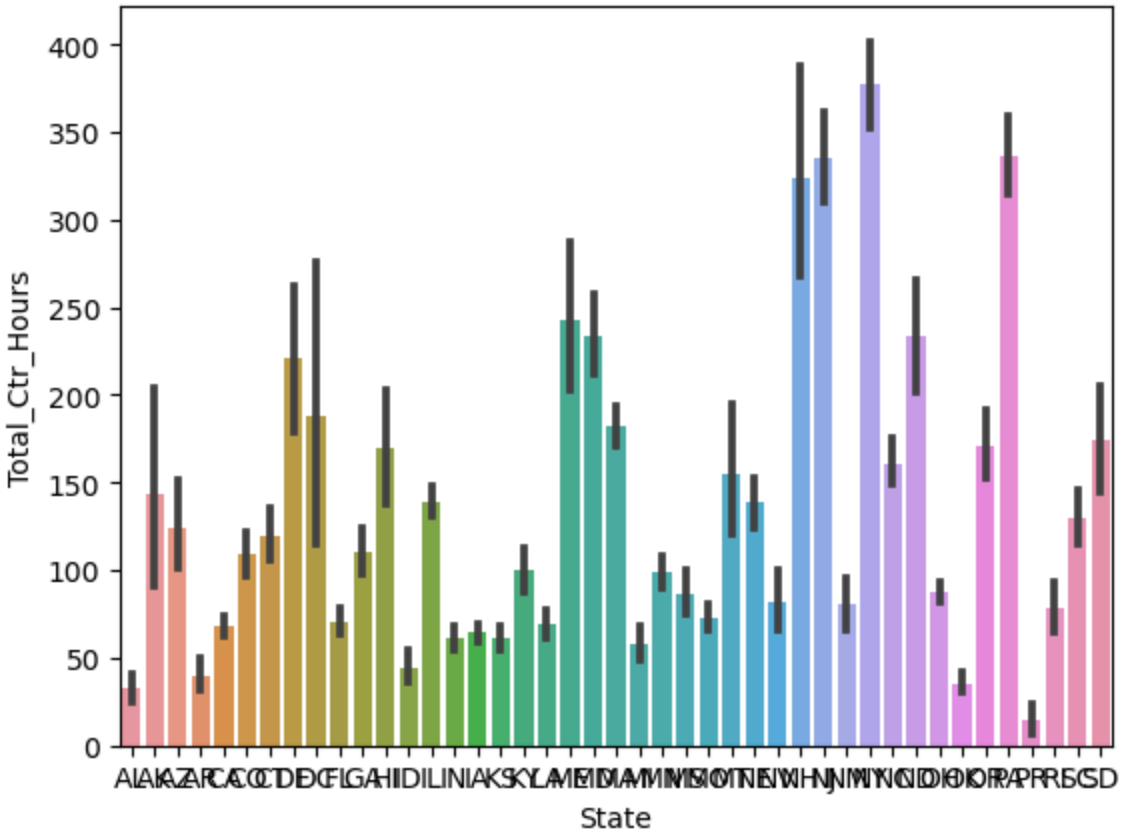
In [12]:
```
merged_df = pd.merge(merged_df, staffing_summary_reset[['PROVNUM', 'Total_Emp_Hours
                     left_on='PROVNUM', right_on='PROVNUM', how='left')
```

In [13]:
```
merged_df.to_csv('merged_df.csv', index=False)
```

In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

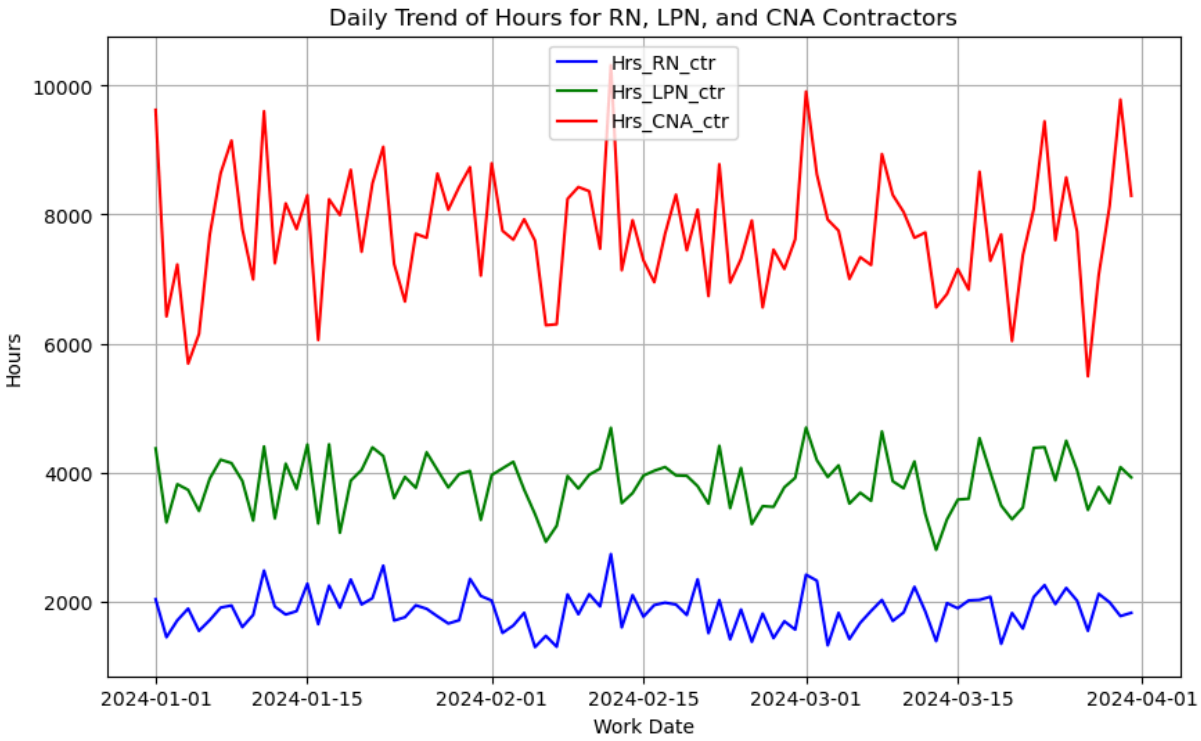
In [15]:
```
sns.barplot(data=merged_df, x='State', y='Total_Ctr_Hours')
```

Out[15]: <Axes: xlabel='State', ylabel='Total_Ctr_Hours'>



In [16]:
```
import matplotlib.pyplot as plt

# Convert 'WorkDate' to datetime format
staffing_data_sample['WorkDate'] = pd.to_datetime(staffing_data_sample['WorkDate'],

# Group by 'WorkDate' and sum the specified columns
daily_trend = staffing_data_sample.groupby('WorkDate')[['Hrs_RN_ctr', 'Hrs_LPN_ctr'

# Plot the trends
plt.figure(figsize=(10, 6))
plt.plot(daily_trend['WorkDate'], daily_trend['Hrs_RN_ctr'], label='Hrs_RN_ctr', co
plt.plot(daily_trend['WorkDate'], daily_trend['Hrs_LPN_ctr'], label='Hrs_LPN_ctr',
plt.plot(daily_trend['WorkDate'], daily_trend['Hrs_CNA_ctr'], label='Hrs_CNA_ctr',

# Add labels and title
plt.xlabel('Work Date')
plt.ylabel('Hours')
plt.title('Daily Trend of Hours for RN, LPN, and CNA Contractors')
plt.legend()
plt.grid(True)
```

```
# Show the plot
plt.show()
```



In [17]:
```
merged_df.dtypes
```

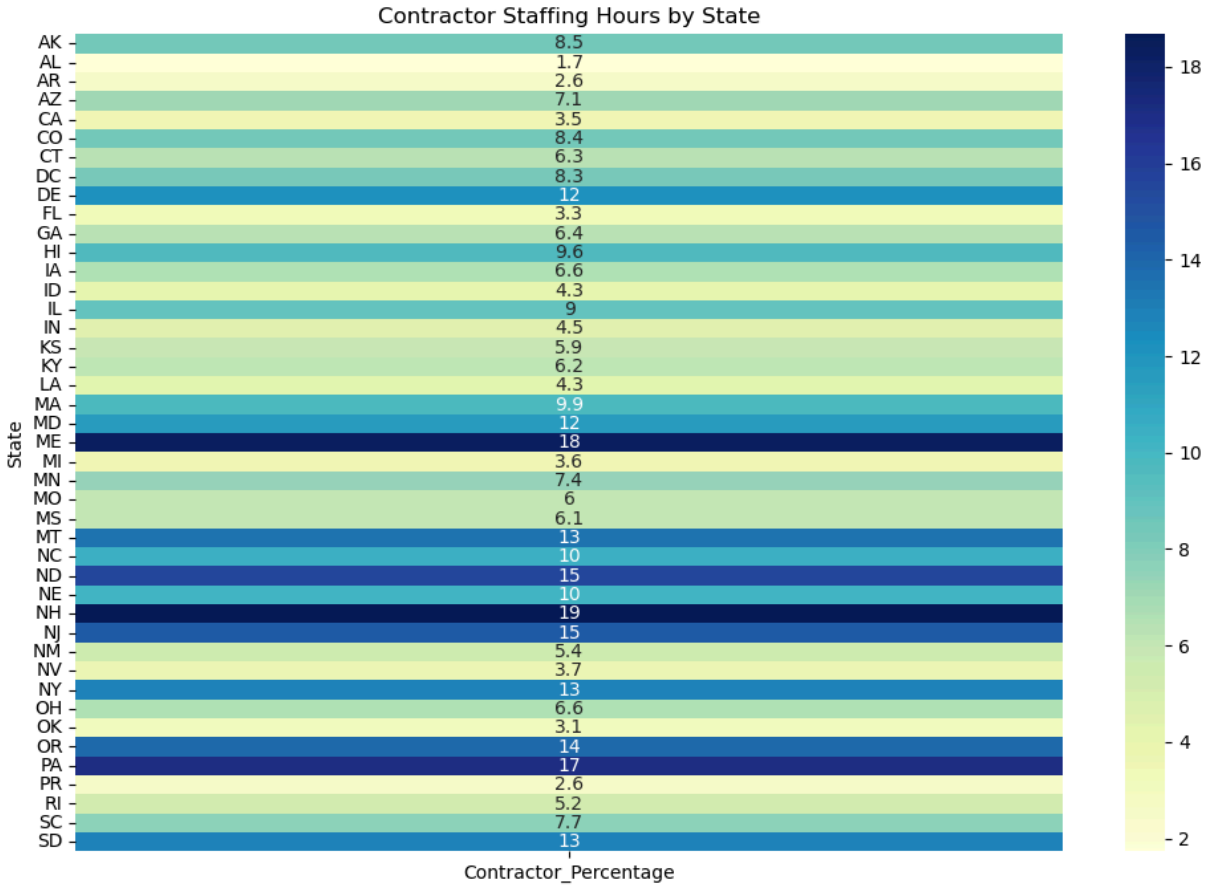
Out[17]:
```
CCN                                                                  object
Provider Name                                                        object
City/Town                                                            object
State                                                                object
County/Parish                                                        object
Ownership Type                                                       object
Number of Certified Beds                                              int64
Provider Type                                                        object
Overall Rating                                                        int64
Staffing Rating                                                       int64
Reported Nurse Aide Staffing Hours per Resident per Day             float64
Reported LPN Staffing Hours per Resident per Day                    float64
Reported RN Staffing Hours per Resident per Day                     float64
Reported Licensed Staffing Hours per Resident per Day               float64
Reported Total Nurse Staffing Hours per Resident per Day            float64
Total number of nurse staff hours per resident per day on the weekend    float64
Registered Nurse hours per resident per day on the weekend          float64
Reported Physical Therapist Staffing Hours per Resident Per Day     float64
PROVNUM                                                              object
WorkDate                                                              int64
Hrs_RNDON                                                           float64
Hrs_RNDON_emp                                                       float64
Hrs_RNDON_ctr                                                       float64
Hrs_RNadmin                                                         float64
Hrs_RNadmin_emp                                                     float64
Hrs_RNadmin_ctr                                                     float64
Hrs_RN                                                              float64
Hrs_RN_emp                                                          float64
Hrs_RN_ctr                                                          float64
Hrs_LPNadmin                                                        float64
Hrs_LPNadmin_emp                                                    float64
Hrs_LPNadmin_ctr                                                    float64
Hrs_LPN                                                             float64
Hrs_LPN_emp                                                         float64
Hrs_LPN_ctr                                                         float64
Hrs_CNA                                                             float64
Hrs_CNA_emp                                                         float64
Hrs_CNA_ctr                                                         float64
Hrs_NAtrn                                                           float64
Hrs_NAtrn_emp                                                       float64
Hrs_NAtrn_ctr                                                       float64
Hrs_MedAide                                                         float64
Hrs_MedAide_emp                                                     float64
Hrs_MedAide_ctr                                                     float64
Total_Emp_Hours                                                     float64
Total_Ctr_Hours                                                     float64
dtype: object
```

In [18]:
```python
## question b
state_summary = merged_df.groupby('State').agg({
    'Total_Emp_Hours': 'sum',
    'Total_Ctr_Hours': 'sum'
}).reset_index()
```

```
# Calculate percentage of staffing hours worked by contractors
state_summary['Contractor_Percentage'] = (state_summary['Total_Ctr_Hours'] / (state
```

In [19]:

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

state_summary.set_index('State', inplace=True)

# Create the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(state_summary[['Contractor_Percentage']], cmap='YlGnBu', annot=True, cb
plt.title('Contractor Staffing Hours by State')
plt.show()
```

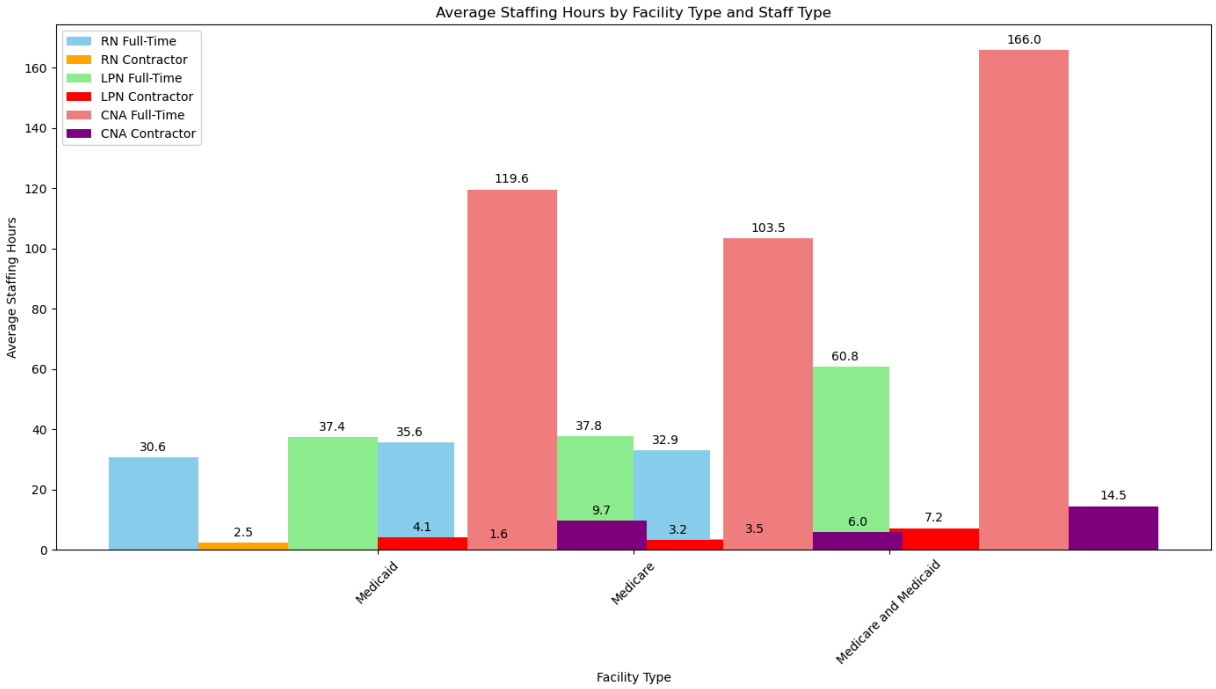


In [20]:

```
## question a
import pandas as pd
import matplotlib.pyplot as plt

# Example data preparation
# Assuming df is your DataFrame

# Group by facility type and calculate the average staffing hours for contractors a
facility_summary = merged_df.groupby('Provider Type').agg({
    'Hrs_RN_emp': 'mean',
    'Hrs_RN_ctr': 'mean',
    'Hrs_LPN_emp': 'mean',
    'Hrs_LPN_ctr': 'mean',
    'Hrs_CNA_emp': 'mean',
```

```
    'Hrs_CNA_ctr': 'mean'
}).reset_index()

# Rename columns for better understanding
facility_summary.columns = [
    'Facility Type',
    'Average RN Hours (Full-Time)',
    'Average RN Hours (Contractor)',
    'Average LPN Hours (Full-Time)',
    'Average LPN Hours (Contractor)',
    'Average CNA Hours (Full-Time)',
    'Average CNA Hours (Contractor)'
]

# Plot bar chart
fig, ax = plt.subplots(figsize=(14, 8))

# Set bar width and position
bar_width = 0.35
index = range(len(facility_summary))

# Plotting
bar1 = ax.bar(index, facility_summary['Average RN Hours (Full-Time)'], bar_width, l
bar2 = ax.bar([i + bar_width for i in index], facility_summary['Average RN Hours (C
bar3 = ax.bar([i + 2*bar_width for i in index], facility_summary['Average LPN Hours
bar4 = ax.bar([i + 3*bar_width for i in index], facility_summary['Average LPN Hours
bar5 = ax.bar([i + 4*bar_width for i in index], facility_summary['Average CNA Hours
bar6 = ax.bar([i + 5*bar_width for i in index], facility_summary['Average CNA Hours

# Adding labels and title
ax.set_xlabel('Facility Type')
ax.set_ylabel('Average Staffing Hours')
ax.set_title('Average Staffing Hours by Facility Type and Staff Type')
ax.set_xticks([i + 2.5*bar_width for i in index])
ax.set_xticklabels(facility_summary['Facility Type'], rotation=45)

# Adding a legend
ax.legend()

# Adding data labels on top of bars
def add_labels(bars):
    for bar in bars:
        height = bar.get_height()
        ax.annotate(f'{height:.1f}',
                    xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

# Apply labels
add_labels(bar1)
add_labels(bar2)
add_labels(bar3)
add_labels(bar4)
add_labels(bar5)
add_labels(bar6)
```

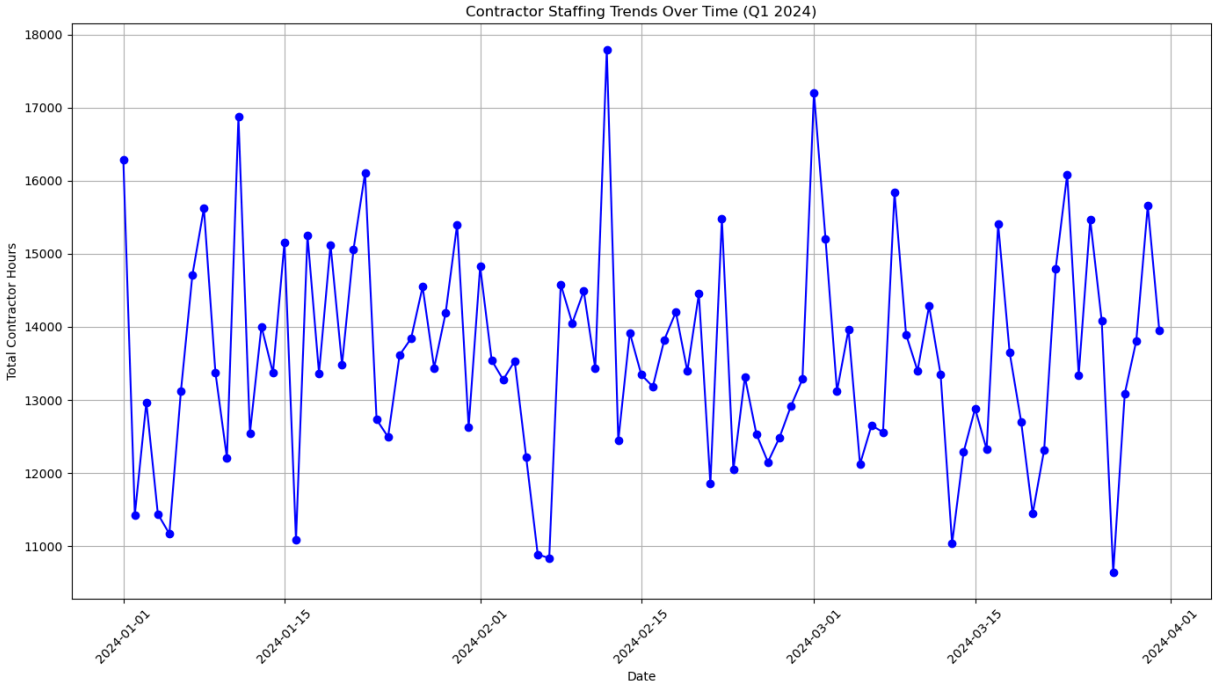
```
plt.tight_layout()
plt.show()
```



In [21]:

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

# Example data preparation
# Assuming df is your DataFrame

# Convert WorkDate to datetime if it's not already
merged_df['WorkDate'] = pd.to_datetime(merged_df['WorkDate'], format='%Y%m%d')

# Filter data for the first quarter of 2024
q1_2024_df = merged_df[(merged_df['WorkDate'] >= '2024-01-01') & (merged_df['WorkDa

# Aggregate contractor hours by day
daily_contractor_hours = q1_2024_df.groupby('WorkDate').agg({
    'Hrs_RNDON_ctr': 'sum',
    'Hrs_RNadmin_ctr': 'sum',
    'Hrs_RN_ctr': 'sum',
    'Hrs_LPNadmin_ctr': 'sum',
    'Hrs_LPN_ctr': 'sum',
    'Hrs_CNA_ctr': 'sum'
}).reset_index()

# Calculate total contractor hours for each day
daily_contractor_hours['Total_Contractor_Hours'] = daily_contractor_hours[['Hrs_RND

# Plot line graph
plt.figure(figsize=(14, 8))
plt.plot(daily_contractor_hours['WorkDate'], daily_contractor_hours['Total_Contract

# Adding labels and title
plt.xlabel('Date')
```

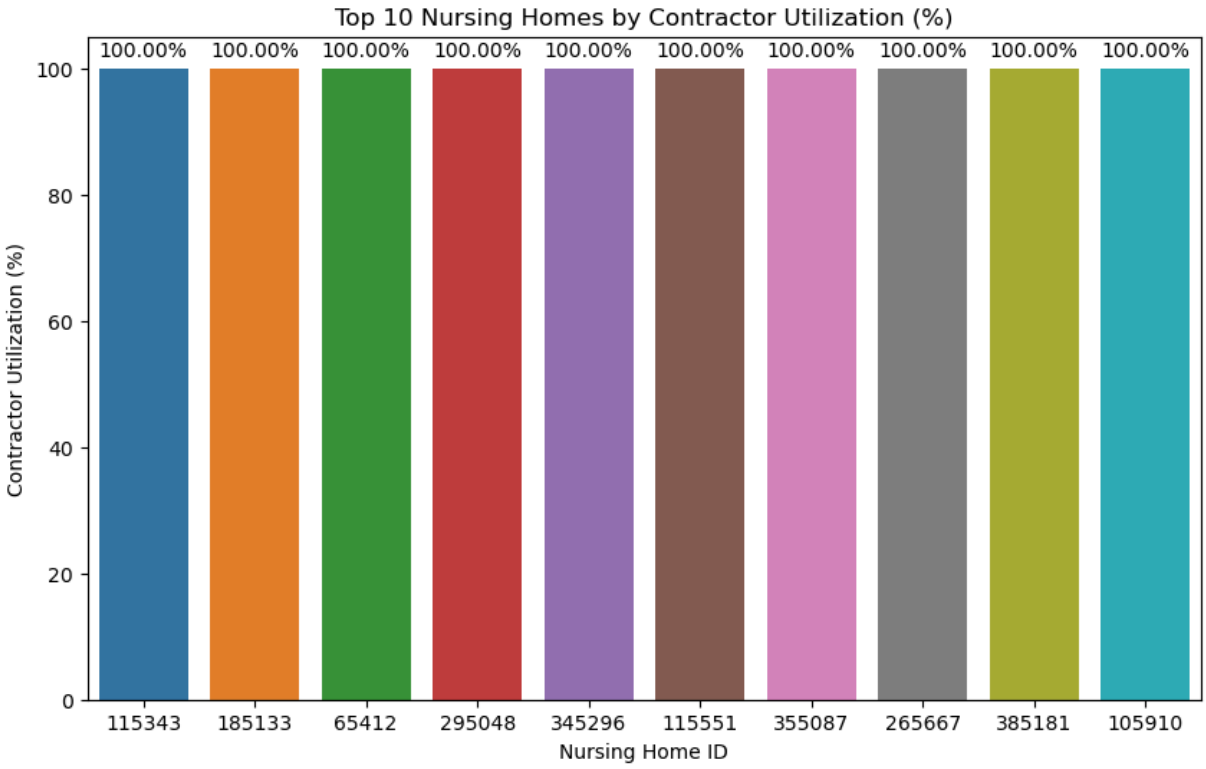
```
plt.ylabel('Total Contractor Hours')
plt.title('Contractor Staffing Trends Over Time (Q1 2024)')
plt.grid(True)

# Rotate date labels for better readability
plt.xticks(rotation=45)

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



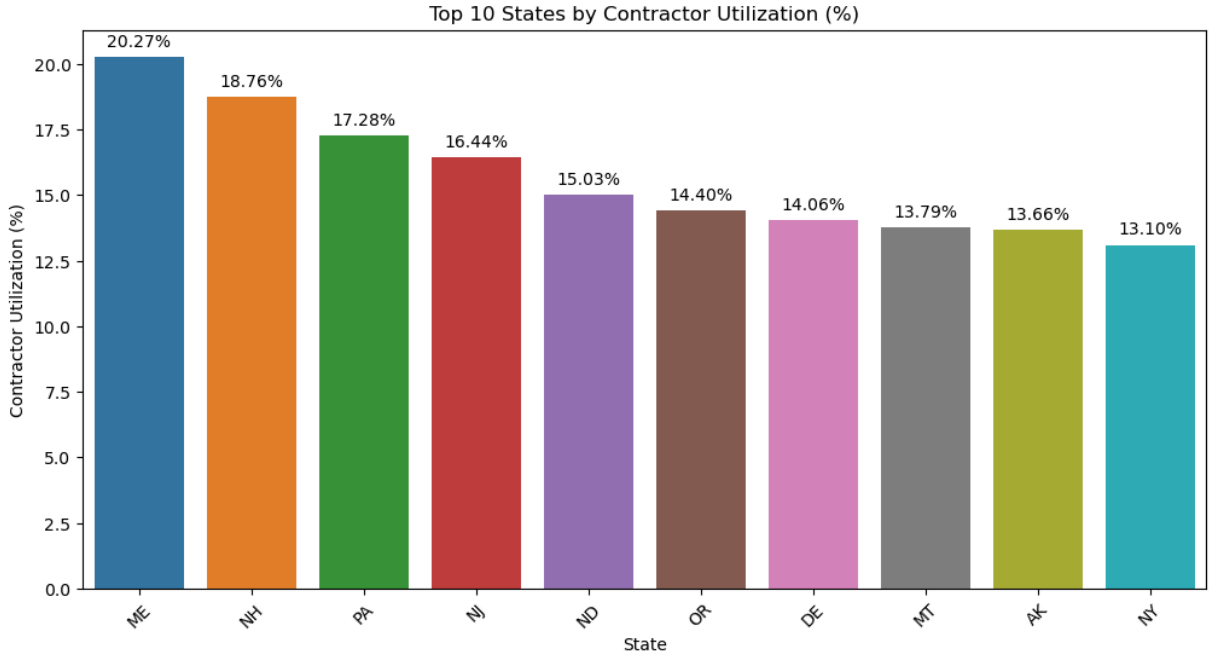
In [22]:
```
##Question 1
# Calculate contractor utilization percentage
contractor_hours = staffing_data_df.groupby('PROVNUM')[['Hrs_RN_ctr', 'Hrs_LPN_ctr'
employee_hours = staffing_data_df.groupby('PROVNUM')[['Hrs_RN_emp', 'Hrs_LPN_emp',

# Merge the contractor and employee hours
merged_df = pd.merge(contractor_hours, employee_hours, on='PROVNUM', how='inner')

# Calculate contractor utilization percentage
merged_df['Contractor Utilization (%)'] = (merged_df['Total Contractor Hours'] /
                                           (merged_df['Total Contractor Hours'] + m

# Display top 10 facilities by contractor utilization
top_facilities = merged_df.sort_values(by='Contractor Utilization (%)', ascending=F
print(top_facilities)

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

plt.figure(figsize=(10, 6))
ax=sns.barplot(x='PROVNUM', y='Contractor Utilization (%)', data=top_facilities)
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f') + '%',  # Format to two decimal place
                (p.get_x() + p.get_width() / 2., p.get_height()),
```

```
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')
plt.title('Top 10 Nursing Homes by Contractor Utilization (%)')
plt.xlabel('Nursing Home ID')
plt.ylabel('Contractor Utilization (%)')
plt.show()
```

```
      PROVNUM  Total Contractor Hours  Total Employee Hours  \
822    115343                3728.45                   0.0
3338   185133                1380.26                   0.0
11252   65412                3503.97                   0.0
6188   295048                5543.00                   0.0
7513   345296               13495.40                   0.0
950    115551               26757.12                   0.0
7919   355087                3104.71                   0.0
5755   265667                 537.67                   0.0
9194   385181                 755.25                   0.0
575    105910               52901.26                   0.0

       Contractor Utilization (%)
822                         100.0
3338                        100.0
11252                       100.0
6188                        100.0
7513                        100.0
950                         100.0
7919                        100.0
5755                        100.0
9194                        100.0
575                         100.0
```



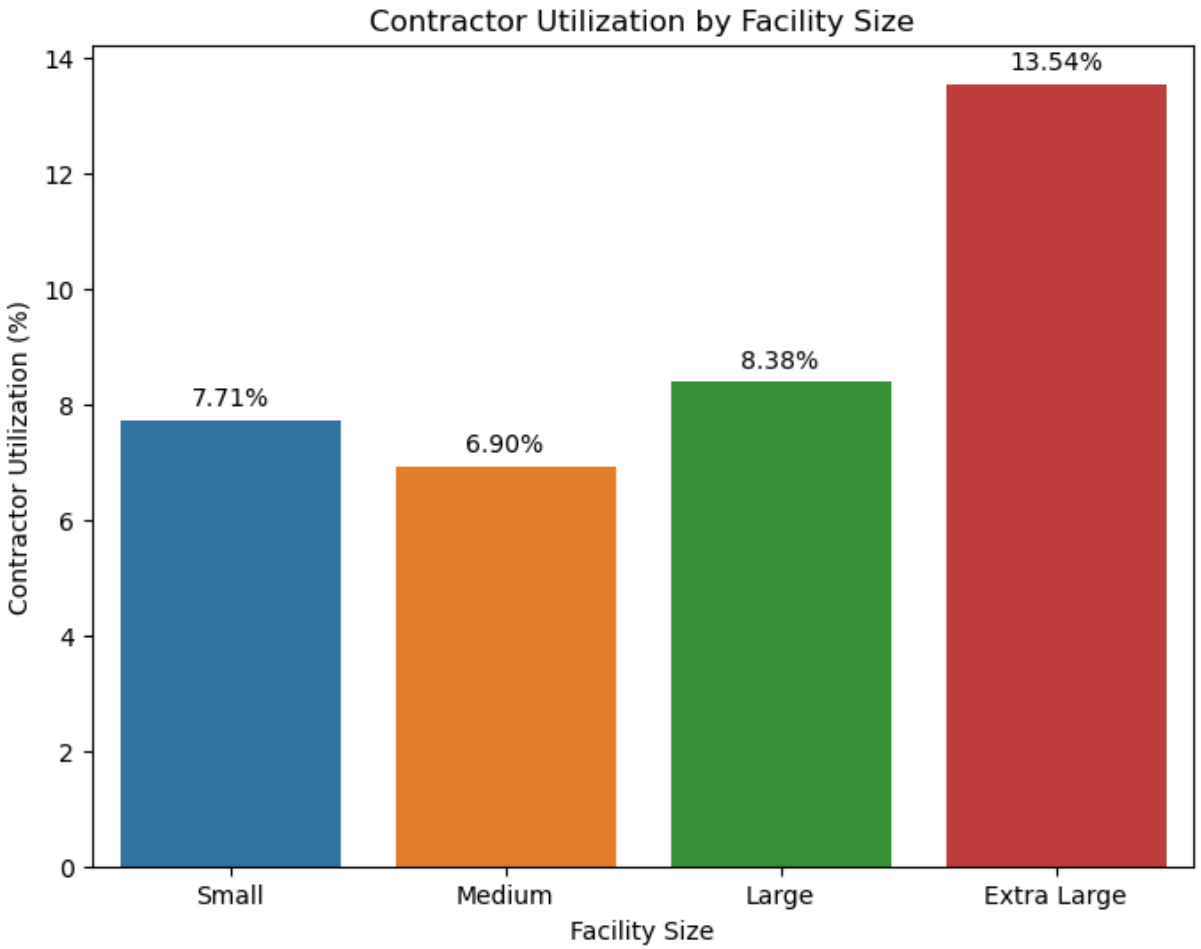
In [23]:
```
## Question2
# Merge with provider info data to get state/region
merged_df = pd.merge(merged_df, provider_info_df[['CCN', 'State']], left_on='PROVNU

# Group by state and calculate total hours
statewise_usage = merged_df.groupby('State').agg({
    'Total Contractor Hours': 'sum',
    'Total Employee Hours': 'sum'
}).reset_index()

# Calculate contractor utilization percentage by state
statewise_usage['Contractor Utilization (%)'] = (statewise_usage['Total Contractor
                                                 (statewise_usage['Total Contractor

# Visualization

plt.figure(figsize=(12, 6))
ax=sns.barplot(x='State', y='Contractor Utilization (%)', data=statewise_usage.sort
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f') + '%',  # Format to two decimal place
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')
plt.title('Top 10 States by Contractor Utilization (%)')
plt.xlabel('State')
plt.ylabel('Contractor Utilization (%)')
plt.xticks(rotation=45)
plt.show()
```



In [24]:
```
## Question 3
# Create a facility size category in the provider info dataset
provider_info_df['Facility Size Category'] = pd.cut(provider_info_df['Number of Cer

# Merge the facility size data with staffing data
```

```
sizewise_usage = pd.merge(merged_df, provider_info_df[['CCN', 'Facility Size Catego

# Group by facility size category
size_grouped = sizewise_usage.groupby('Facility Size Category').agg({
    'Total Contractor Hours': 'sum',
    'Total Employee Hours': 'sum'
}).reset_index()

# Calculate contractor utilization percentage by facility size
size_grouped['Contractor Utilization (%)'] = (size_grouped['Total Contractor Hours'
                                              (size_grouped['Total Contractor Hours

# Visualization
plt.figure(figsize=(8, 6))
ax=sns.barplot(x='Facility Size Category', y='Contractor Utilization (%)', data=siz
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f') + '%',  # Format to two decimal place
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')
plt.title('Contractor Utilization by Facility Size')
plt.xlabel('Facility Size')
plt.ylabel('Contractor Utilization (%)')
plt.show()
```

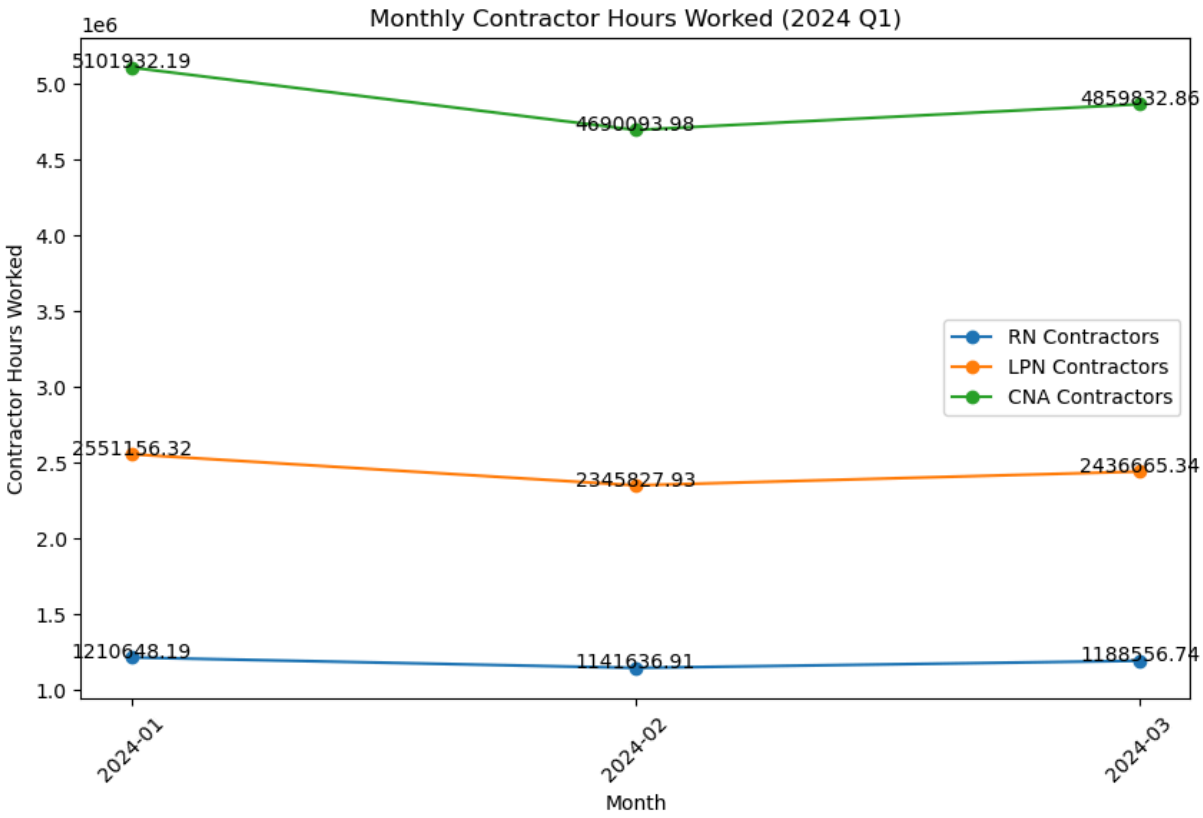
```
C:\Users\niral\AppData\Local\Temp\ipykernel_3320\3234750201.py:9: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future vers
ion of pandas. Pass observed=False to retain current behavior or observed=True to ad
opt the future default and silence this warning.
  size_grouped = sizewise_usage.groupby('Facility Size Category').agg({
C:\Users\niral\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=Tru
e to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



In [25]:

```
## Question 4
# Convert 'WorkDate' to datetime
staffing_data_df['WorkDate'] = pd.to_datetime(staffing_data_df['WorkDate'], format=

# Group by WorkDate and sum contractor hours
monthly_trends = staffing_data_df.groupby(staffing_data_df['WorkDate'].dt.to_period
    'Hrs_RN_ctr': 'sum',
    'Hrs_LPN_ctr': 'sum',
    'Hrs_CNA_ctr': 'sum'
}).reset_index()

# Plotting the trend
plt.figure(figsize=(10, 6))
plt.plot(monthly_trends['WorkDate'].astype(str), monthly_trends['Hrs_RN_ctr'], labe
plt.plot(monthly_trends['WorkDate'].astype(str), monthly_trends['Hrs_LPN_ctr'], lab
plt.plot(monthly_trends['WorkDate'].astype(str), monthly_trends['Hrs_CNA_ctr'], lab

# Add labels to the line graph points
for i in range(monthly_trends.shape[0]):
    plt.text(i, monthly_trends['Hrs_RN_ctr'].iloc[i], f"{monthly_trends['Hrs_RN_ctr
    plt.text(i, monthly_trends['Hrs_LPN_ctr'].iloc[i], f"{monthly_trends['Hrs_LPN_c
    plt.text(i, monthly_trends['Hrs_CNA_ctr'].iloc[i], f"{monthly_trends['Hrs_CNA_c


plt.title('Monthly Contractor Hours Worked (2024 Q1)')
plt.xlabel('Month')
```

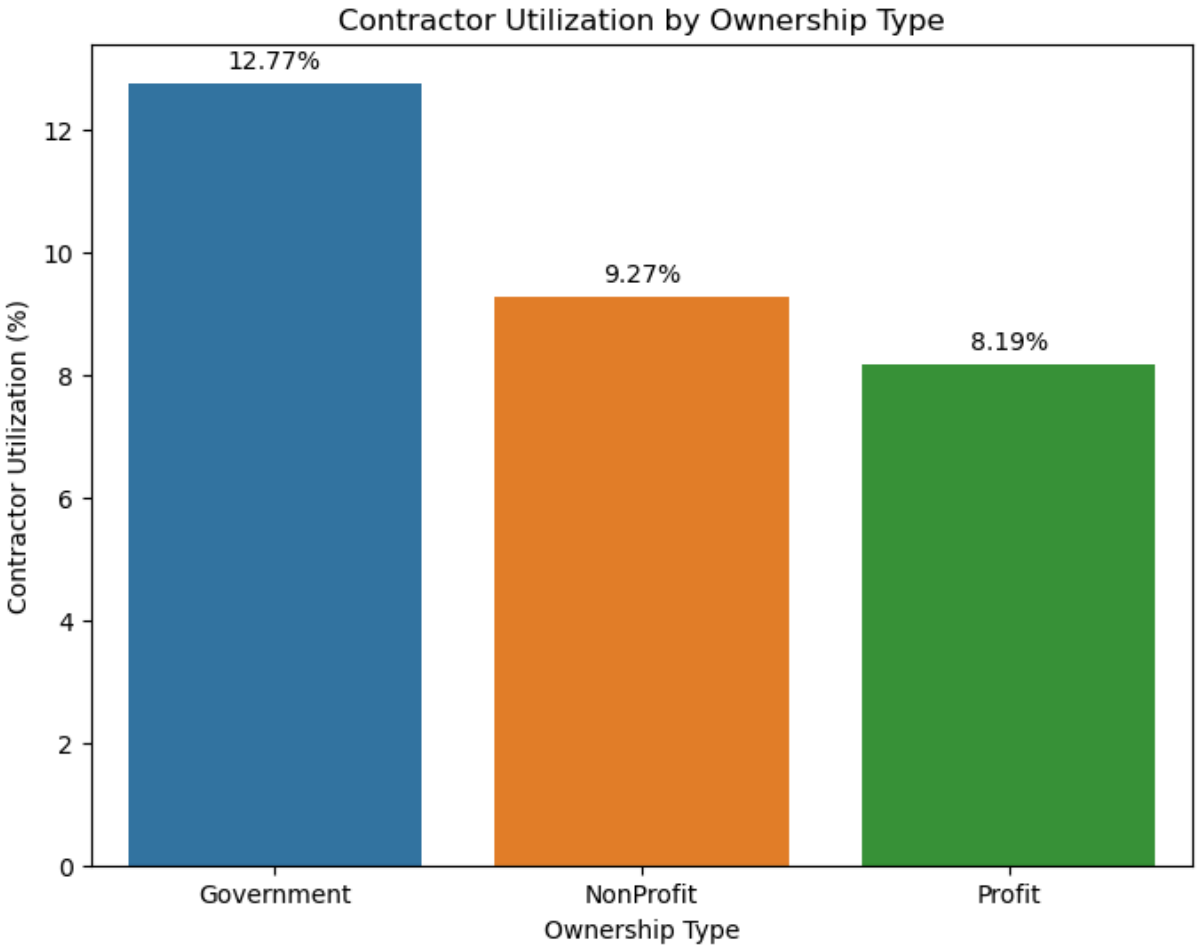
```
plt.ylabel('Contractor Hours Worked')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```



In [26]:

```
## question 5
# Group by ownership type and calculate total hours
ownership_usage = pd.merge(merged_df, provider_info_df[['CCN', 'Ownership Type']],

ownership_grouped = ownership_usage.groupby('Ownership Type').agg({
    'Total Contractor Hours': 'sum',
    'Total Employee Hours': 'sum'
}).reset_index()

# Calculate contractor utilization percentage by ownership type
ownership_grouped['Contractor Utilization (%)'] = (ownership_grouped['Total Contrac
                                                   (ownership_grouped['Total Contra

# Visualization
plt.figure(figsize=(8, 6))
ax=sns.barplot(x='Ownership Type', y='Contractor Utilization (%)', data=ownership_g
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f') + '%',  # Format to two decimal place
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')
plt.title('Contractor Utilization by Ownership Type')
plt.xlabel('Ownership Type')
```

```
plt.ylabel('Contractor Utilization (%)')
plt.show()
```



In [32]:
```
merged_df['Contractor Utilization (%)'] = (merged_df['Total Contractor Hours'] /
                                           (merged_df['Total Contractor Hours'] + m

# Now, calculate summary statistics for contractor utilization
sample_mean_contractor_utilization = merged_df['Contractor Utilization (%)'].mean()
sample_median_contractor_utilization = merged_df['Contractor Utilization (%)'].medi
sample_std_contractor_utilization = merged_df['Contractor Utilization (%)'].std()

print(f"Sample Mean Contractor Utilization: {sample_mean_contractor_utilization:.2f
print(f"Sample Median Contractor Utilization: {sample_median_contractor_utilization
print(f"Sample Standard Deviation of Contractor Utilization: {sample_std_contractor
```

```
Sample Mean Contractor Utilization: 7.97
Sample Median Contractor Utilization: 1.19
Sample Standard Deviation of Contractor Utilization: 12.84
```

In [37]:
```
# Calculate summary statistics for the full dataset
staffing = staffing_data_df.groupby('PROVNUM').agg(
    total_RN_emp=('Hrs_RN_emp', 'sum'),
    total_RN_ctr=('Hrs_RN_ctr', 'sum'),
    total_LPN_emp=('Hrs_LPN_emp', 'sum'),
    total_LPN_ctr=('Hrs_LPN_ctr', 'sum'),
    total_CNA_emp=('Hrs_CNA_emp', 'sum'),
    total_CNA_ctr=('Hrs_CNA_ctr', 'sum'),
```

```
).reset_index()
# Summing the total employee and contractor hours across all roles (RN, LPN, CNA, e
staffing['Total_Emp_Hours'] = staffing.filter(like='_emp').sum(axis=1)
staffing['Total_Ctr_Hours'] = staffing.filter(like='_ctr').sum(axis=1)



staffing['Contractor Utilization (%)'] = (staffing['Total_Ctr_Hours'] /
                                          (staffing['Total_Ctr_Hours'] + staffing[

full_mean_contractor_utilization = staffing['Contractor Utilization (%)'].mean()
full_median_contractor_utilization = staffing['Contractor Utilization (%)'].median(
full_std_contractor_utilization = staffing['Contractor Utilization (%)'].std()

# Print summary statistics for the full dataset with 2 decimal places
print(f"Full Dataset Mean Contractor Utilization: {full_mean_contractor_utilization
print(f"Full Dataset Median Contractor Utilization: {full_median_contractor_utiliza
print(f"Full Dataset Standard Deviation of Contractor Utilization: {full_std_contra
```

```
Full Dataset Mean Contractor Utilization: 7.97
Full Dataset Median Contractor Utilization: 1.19
Full Dataset Standard Deviation of Contractor Utilization: 12.84
```

In [ ]: