Operating_room_utilization_by_Diptyajit_Das

June 9, 2024

0.1 Problem statement

Operating room (OR) inefficiency is a significant financial burden on healthcare organizations, impacting both cost and patient care. While booked OR time represents a planned utilization metric, it often deviates from the actual time procedures take due to workflow delays, inaccurate booking estimates, and cancellations. This project aims to leverage a dataset containing surgical timestamps throughout the OR workflow to identify and quantify these areas of inefficiency. By analyzing this data, we can develop actionable insights to optimize OR utilization, potentially saving healthcare organizations substantial time and financial resources, and ultimately improving patient care delivery.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import levene,ttest_ind
import warnings
warnings.simplefilter('ignore')
```

[2]:
```
df=pd.read_csv('Operating_room_utilization_dataset.csv',index_col=0)
df=df.rename(columns={'Booked Time (min)':'Booked Time'})
df.shape
```

[2]:
```
(2172, 12)
```

0.2 2172 rows and 12 columns

[3]:
```
df.isna().sum()
```

[3]:
```
Encounter ID       0
Date               0
OR Suite           0
Service            0
CPT Code           0
CPT Description    0
Booked Time        0
OR Schedule        0
Wheels In          0
Start Time         0
```

```
End Time          0
Wheels Out        0
dtype: int64
```

0.3 No missing values.

[4]:
```
df[df.duplicated()]
```

[4]:
```
Empty DataFrame
Columns: [Encounter ID, Date, OR Suite, Service, CPT Code, CPT Description,
Booked Time, OR Schedule, Wheels In, Start Time, End Time, Wheels Out]
Index: []
```

0.4 No duplicated rows.

0.5 Converting to appropriate datatypes.

[5]:
```
for col in ['Date','OR Schedule','Wheels In','Start Time','End Time','Wheels Out']:
    df[col]=pd.to_datetime(df[col])
for col in ['Encounter ID','OR Suite','CPT Code']:
    df[col]=df[col].astype('object')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2172 entries, 0 to 2171
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Encounter ID     2172 non-null   object
 1   Date             2172 non-null   datetime64[ns]
 2   OR Suite         2172 non-null   object
 3   Service          2172 non-null   object
 4   CPT Code         2172 non-null   object
 5   CPT Description  2172 non-null   object
 6   Booked Time      2172 non-null   int64
 7   OR Schedule      2172 non-null   datetime64[ns]
 8   Wheels In        2172 non-null   datetime64[ns]
 9   Start Time       2172 non-null   datetime64[ns]
 10  End Time         2172 non-null   datetime64[ns]
 11  Wheels Out       2172 non-null   datetime64[ns]
dtypes: datetime64[ns](6), int64(1), object(5)
memory usage: 220.6+ KB
```

0.6 After Converting to Appropriate Datetime and Object Types, the Columns Are:

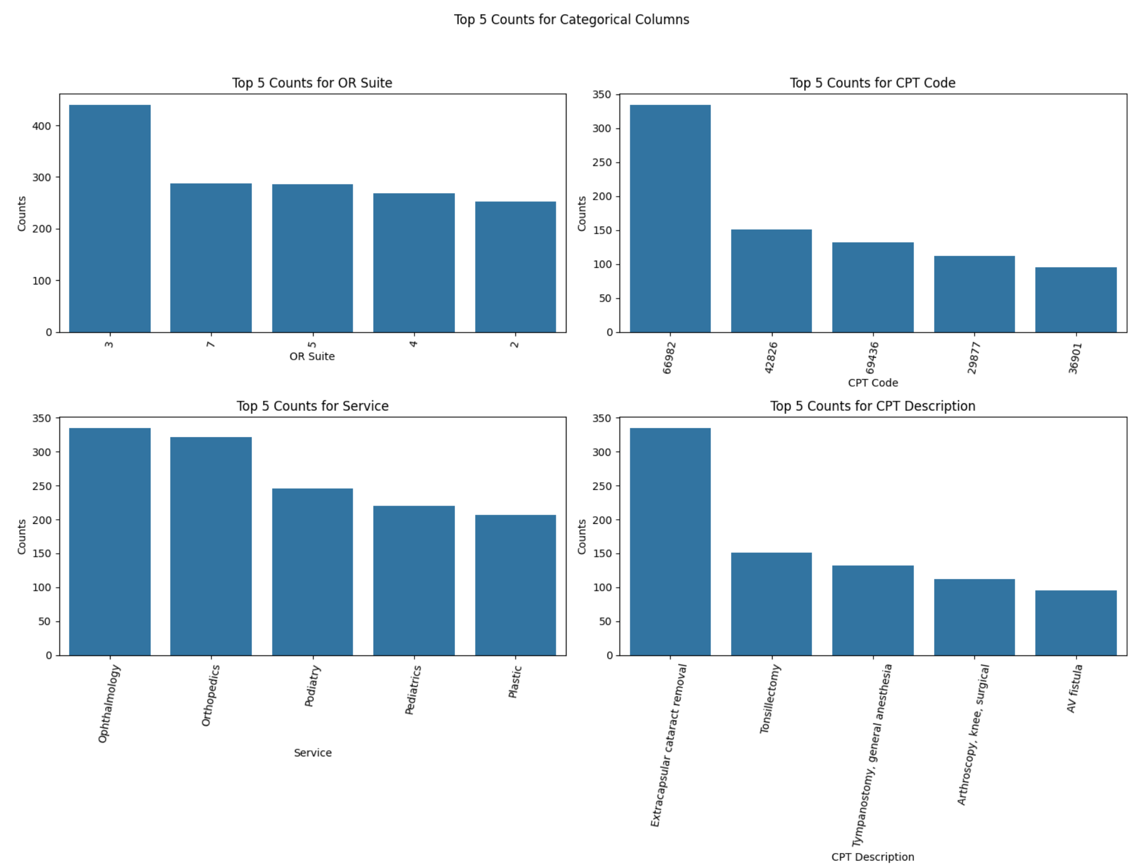
- **Integer**:

    - Booked Time
- **Object**:
    - Encounter ID
    - OR Suite
    - CPT Code
    - Service
    - CPT Description
- **Datetime**:
    - Date
    - OR Schedule
    - Wheels In
    - Start Time
    - End Time
    - Wheels Out

[6]:
```
categorical_columns = ['OR Suite', 'CPT Code', 'Service', 'CPT Description']

fig, axes = plt.subplots(2, 2, figsize=(15, 12))
fig.suptitle('Top 5 Counts for Categorical Columns')

axes = axes.flatten()

for i, col in enumerate(categorical_columns):
    top_5 = df[col].value_counts().nlargest(5)
    print(f'Top 5 Counts for {col} :{top_5}')

    sns.countplot(x=col, data=df, order=top_5.index, ax=axes[i])
    axes[i].set_title(f'Top 5 Counts for {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Counts')
    axes[i].tick_params(axis='x', rotation=80)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

```
Top 5 Counts for OR Suite :OR Suite
3    439
7    288
5    286
4    268
2    252
Name: count, dtype: int64
Top 5 Counts for CPT Code :CPT Code
66982    334
42826    151
69436    132
29877    112
36901     95
```

```
Name: count, dtype: int64
Top 5 Counts for Service :Service
Ophthalmology    334
Orthopedics      321
Podiatry         246
Pediatrics       220
Plastic          207
Name: count, dtype: int64
Top 5 Counts for CPT Description :CPT Description
Extracapsular cataract removal      334
Tonsillectomy                       151
Tympanostomy, general anesthesia    132
Arthroscopy, knee, surgical         112
AV fistula                           95
Name: count, dtype: int64
```



0.6.1 All units of datetime column extractions is minute if not mentioned otherwise.

0.7 Required columns to explain the flow in OR procedures.

- **Booked Time**: The scheduled duration for the use of OR.

- **start__delay**: The delay between the scheduled start time and the actual start time.
- **pre__time**: Time spent on preoperative procedures.
- **OR__time**: Actual time spent for surgery.
- **post__time**: Time spent on postoperative procedures.
- **total__time**: The total time from the start to the end of the entire surgical process.
- **diff**: The difference of total time and booked time.

[7]:
```
df['month']=df['Date'].dt.month
df['week']=df['Date'].dt.isocalendar().week
df['OR_schedule_hour']=df['OR Schedule'].dt.hour
df['start_delay']=(df['Start Time']-df['OR Schedule']).dt.total_seconds() / 60
df['OR_time']=(df['End Time']-df['Start Time']).dt.total_seconds() / 60
df['total_time']=(df['Wheels Out']-df['Wheels In']).dt.total_seconds() / 60
df['pre_time']=(df['Start Time']-df['Wheels In']).dt.total_seconds() / 60
df['post_time']=(df['Wheels Out']-df['End Time']).dt.total_seconds() / 60
df['diff']=df['total_time']-df['Booked Time']
df
```

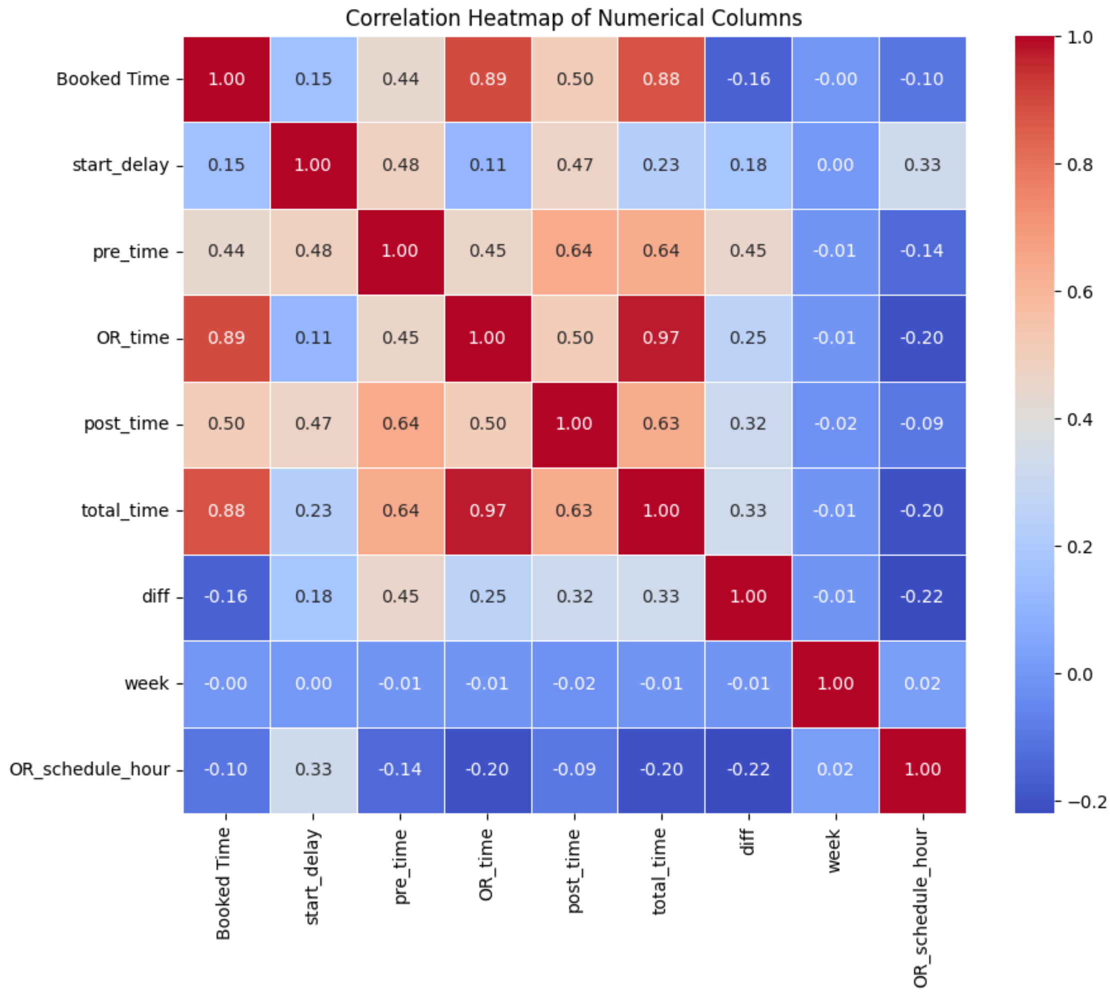
[7]:
```
       Encounter ID       Date OR Suite      Service CPT Code  \
index
0             10001 2022-01-03        1     Podiatry    28110
1             10002 2022-01-03        1     Podiatry    28055
2             10003 2022-01-03        1     Podiatry    28297
3             10004 2022-01-03        1     Podiatry    28296
4             10005 2022-01-03        2  Orthopedics    27445
…                 …          …        …            …        …
2167          12168 2022-03-31        7   Pediatrics    69421
2168          12169 2022-03-31        7   Pediatrics    69421
2169          12170 2022-03-31        8  Orthopedics    27445
2170          12171 2022-03-31        8  Orthopedics    27445
2171          12172 2022-03-31        8  Orthopedics    27130

                                 CPT Description  Booked Time  \
index
0        Partial ostectomy, fifth metatarsal head           90
1       Neurectomy, intrinsic musculature of foot           60
2                            Lapidus bunionectomy          150
3             Bunionectomy with distal osteotomy          120
4             Arthroplasty, knee, hinge prothesis          120
…                                              …            …
2167               Myringotomy, general anesthesia          60
2168               Myringotomy, general anesthesia          60
2169          Arthroplasty, knee, hinge prothesis          120
2170          Arthroplasty, knee, hinge prothesis          120
2171                             Arthroplasty, hip          120

               OR Schedule           Wheels In          Start Time  …  \
```

```
index                                                                     …
0      2022-01-03 07:00:00 2022-01-03 07:05:00 2022-01-03 07:32:00  …
1      2022-01-03 08:45:00 2022-01-03 09:48:00 2022-01-03 10:13:00  …
2      2022-01-03 10:00:00 2022-01-03 11:50:00 2022-01-03 12:20:00  …
3      2022-01-03 12:45:00 2022-01-03 13:29:00 2022-01-03 13:53:00  …
4      2022-01-03 07:00:00 2022-01-03 07:15:00 2022-01-03 07:50:00  …
…                    …                  …                  …   …
2167   2022-03-31 10:45:00 2022-03-31 11:59:00 2022-03-31 12:11:00  …
2168   2022-03-31 12:00:00 2022-03-31 13:20:00 2022-03-31 13:47:00  …
2169   2022-03-31 07:00:00 2022-03-31 07:06:00 2022-03-31 07:45:00  …
2170   2022-03-31 09:15:00 2022-03-31 09:40:00 2022-03-31 10:15:00  …
2171   2022-03-31 11:30:00 2022-03-31 12:40:00 2022-03-31 13:12:00  …

               Wheels Out month  week  OR_schedule_hour  start_delay  OR_time  \
index
0      2022-01-03 09:17:00     1     1                 7         32.0     93.0
1      2022-01-03 11:12:00     1     1                 8         88.0     48.0
2      2022-01-03 12:58:00     1     1                10        140.0     22.0
3      2022-01-03 15:02:00     1     1                12         68.0     57.0
4      2022-01-03 09:51:00     1     1                 7         50.0    108.0
…                    …   …    …                …            …        …
2167   2022-03-31 12:51:00     3    13                10         86.0     28.0
2168   2022-03-31 14:28:00     3    13                12        107.0     27.0
2169   2022-03-31 09:18:00     3    13                 7         45.0     81.0
2170   2022-03-31 12:01:00     3    13                 9         60.0     85.0
2171   2022-03-31 14:58:00     3    13                11        102.0     88.0

       total_time  pre_time  post_time  diff
index
0           132.0      27.0       12.0  42.0
1            84.0      25.0       11.0  24.0
2            68.0      30.0       16.0 -82.0
3            93.0      24.0       12.0 -27.0
4           156.0      35.0       13.0  36.0
…             …        …          …    …
2167         52.0      12.0       12.0  -8.0
2168         68.0      27.0       14.0   8.0
2169        132.0      39.0       12.0  12.0
2170        141.0      35.0       21.0  21.0
2171        138.0      32.0       18.0  18.0

[2172 rows x 21 columns]
```

[8]:
```
numerical=['Booked␣
↪Time','start_delay','pre_time','OR_time','post_time','total_time','diff','week','OR_schedul
numerical_df = df[numerical]
numerical_df.describe()
```

[8]:
```
       Booked Time  start_delay     pre_time      OR_time    post_time  \
count  2172.000000  2172.000000  2172.000000  2172.000000  2172.000000
mean     77.189227    57.072744    21.529926    45.475138    12.691989
std      30.430015    40.602944     6.416851    26.742297     2.667420
min      30.000000   -51.000000     3.000000    12.000000     3.000000
25%      60.000000    28.000000    18.000000    28.000000    11.000000
50%      60.000000    48.000000    23.000000    35.000000    13.000000
75%      90.000000    80.000000    25.000000    58.000000    14.000000
max     180.000000   230.000000    45.000000   136.000000    21.000000

        total_time         diff       week  OR_schedule_hour
count  2172.000000  2172.000000     2172.0       2172.000000
mean     79.697053     2.507827   6.996777          9.414365
std      31.822390    15.364583   3.711311          1.962461
min      19.000000   -82.000000        1.0          7.000000
25%      62.000000    -7.000000        4.0          8.000000
50%      73.000000     3.000000        7.0          9.000000
75%      96.000000    13.000000       10.0         11.000000
max     173.000000    42.000000       13.0         15.000000
```

[9]:
```
correlation_matrix = numerical_df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',
 ↪linewidths=0.5)
plt.title('Correlation Heatmap of Numerical Columns')
plt.show()
```



0.8 Correlation Coefficients Insights and Comments

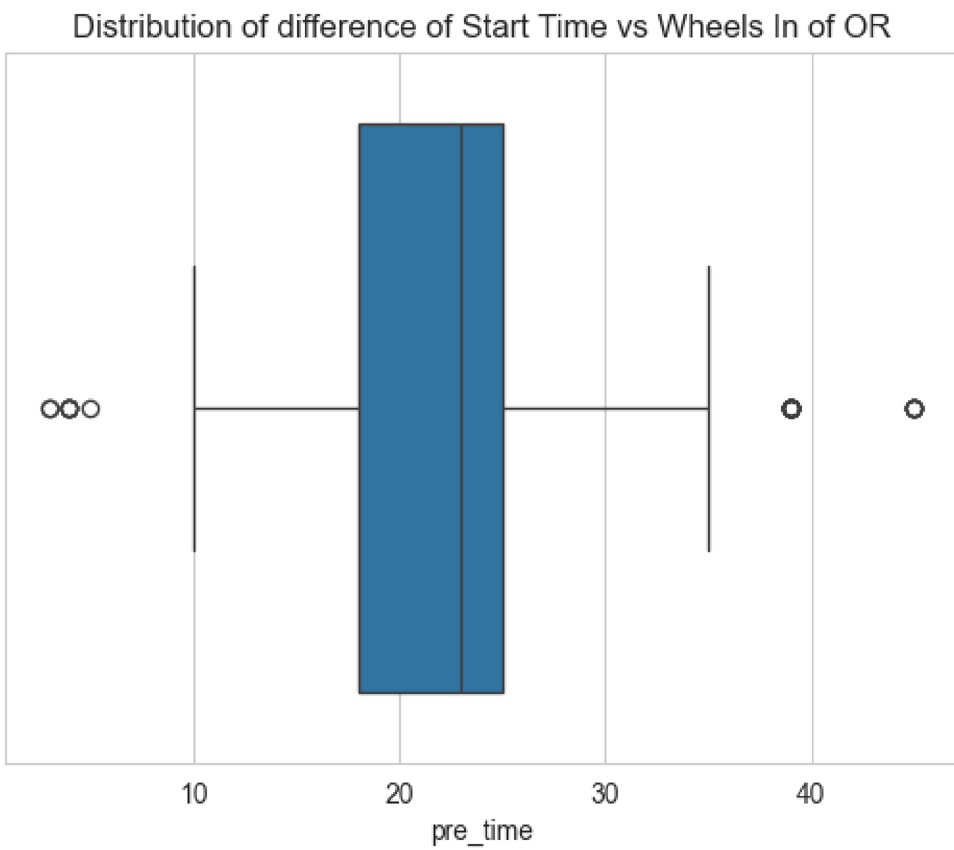
- **High Correlation (0.85)**:
  - **Booked Time and OR Time (0.89)**: The strong correlation indicates that the scheduled time is a good predictor of the actual surgical time, reflecting accurate booking practices or consistent procedure durations.
  - **Booked Time and Total Time (0.88)**: The total time in the OR process is heavily influenced by the initial time booked, suggesting that improvements in booking accuracy could lead to better overall time management.
  - **OR Time and Total Time (0.97)**: The surgical procedure time is the main contributor to the total time, emphasizing the importance of focusing on surgical efficiency to reduce overall OR time.
- **Moderate Correlation (0.5 - 0.7)**:
  - **Preoperative and Postoperative Times (0.64)**: There is a moderate relationship between preoperative and postoperative times, suggesting that delays or efficiencies in one area could impact the other.

  - **Preoperative Time and Total Time (0.64)**: Preoperative activities moderately impact the total OR time.
  - **Postoperative Time and Total Time (0.63)**: Postoperative activities also moderately contribute to the total OR time.
  - **Postoperative Time with Booked Time and OR Time (0.50 each)**: Postoperative time has a moderate correlation with both booked and actual OR times, indicating that the duration of postoperative activities might be influenced by the planned and actual surgery times.

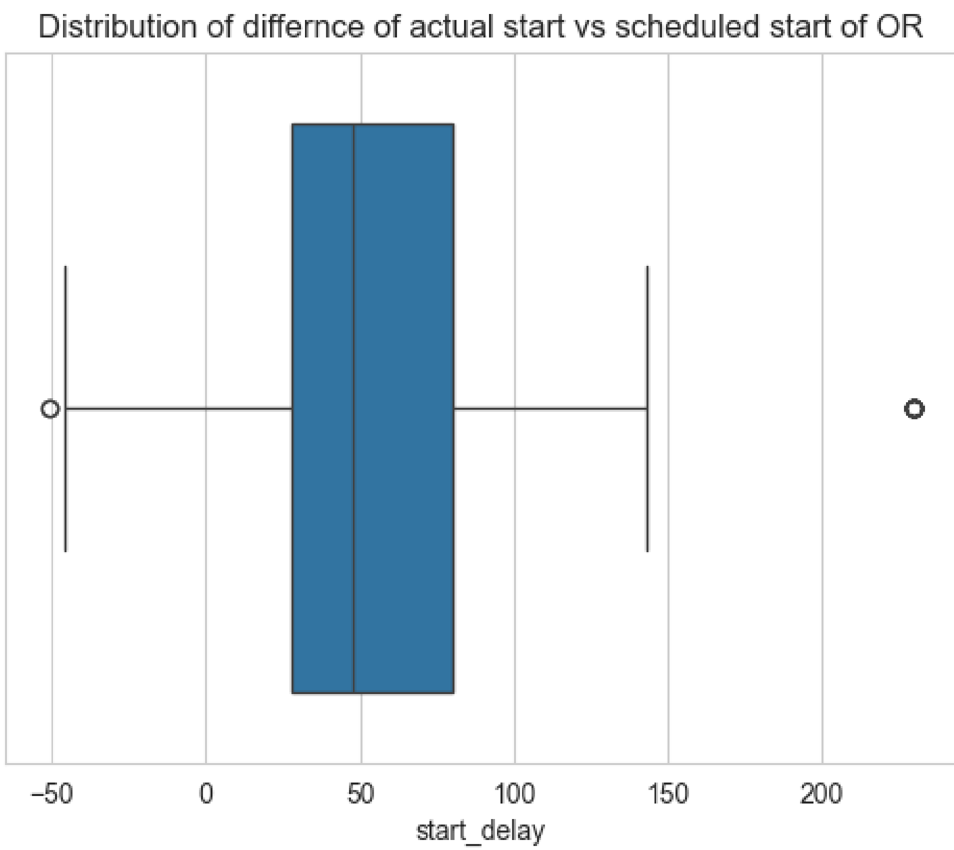
0.9 Conclusion

- The analysis reveals that **OR Time** and **Total Time** are heavily correlated (0.97), indicating that the actual time spent on the surgical procedure is the primary determinant of the total OR time.
  - Though it suggests that **OR Time** reduction will reduce **Total Time** we should give the proper surgery first and then maybe try to reduce time by reducing **pre_time** and **post_time** through technology and better trained staffs.
- The strong correlations of **Booked Time** with both **OR Time** (0.89) and **Total Time** (0.88) highlight the need of updating the **Booked Time** from learning the **OR Time**, **Total Time** and **start_delay** over time.
  - Accurate booking practices are crucial for efficient OR management and resource allocation.

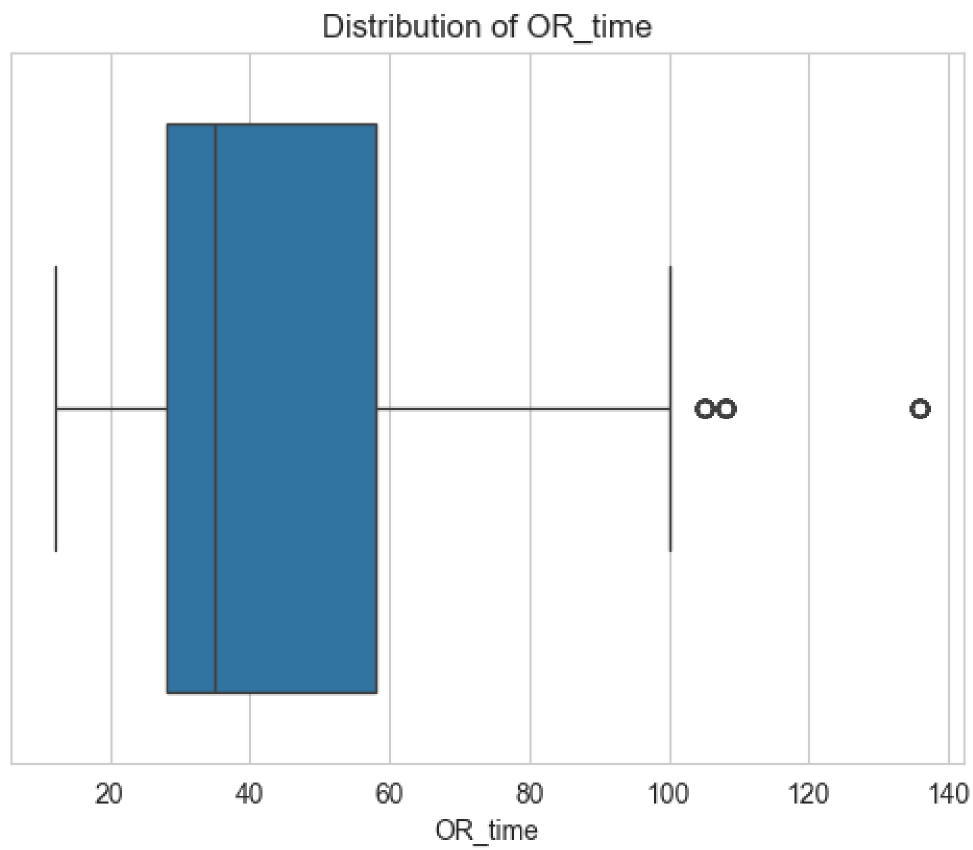
[10]:
```
sns.set_style('whitegrid')
sns.boxplot(data=df,x='pre_time')
plt.title('Distribution of difference of Start Time vs Wheels In of OR')
plt.show()
```



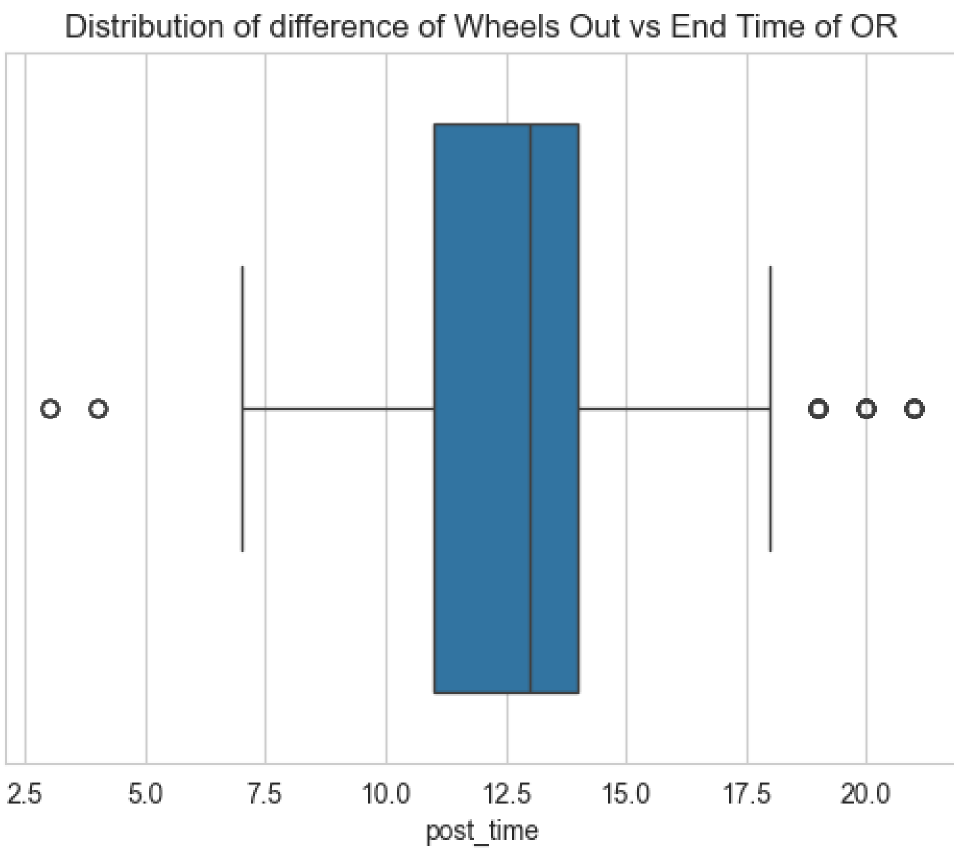

```
[11]: sns.set_style('whitegrid')
      sns.boxplot(data=df,x='start_delay')
      plt.title('Distribution of differnce of actual start vs scheduled start of OR')
      plt.show()
```



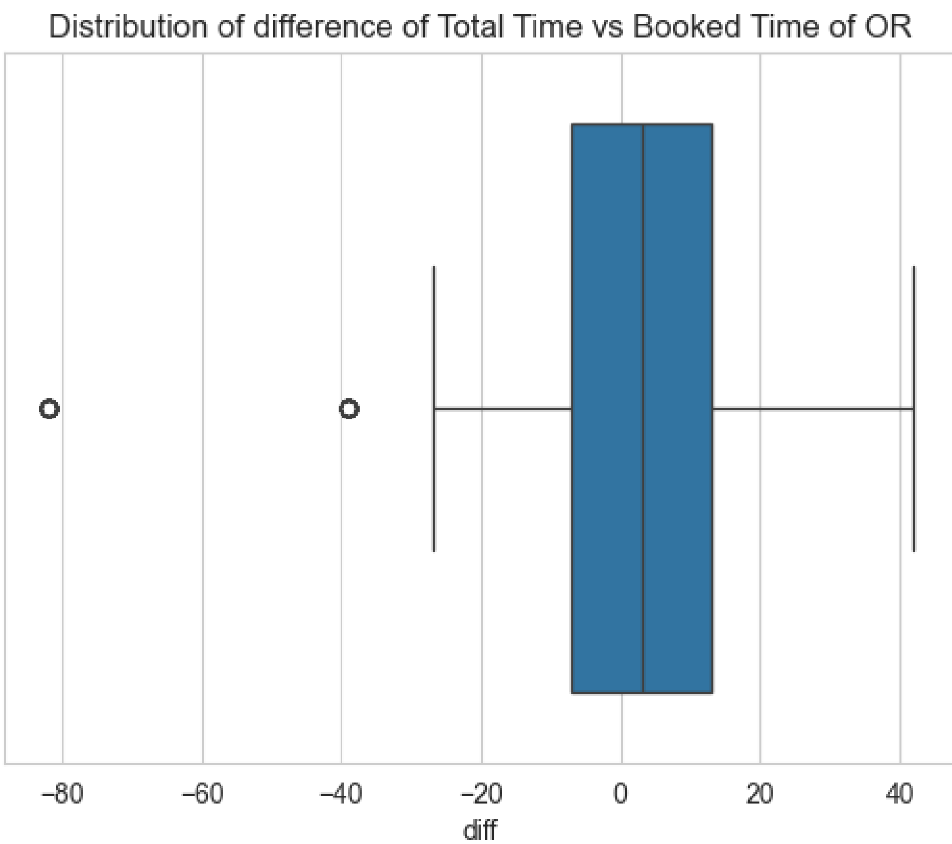
[12]:
```
sns.set_style('whitegrid')
sns.boxplot(data=df,x='OR_time')
plt.title('Distribution of OR_time')
plt.show()
```



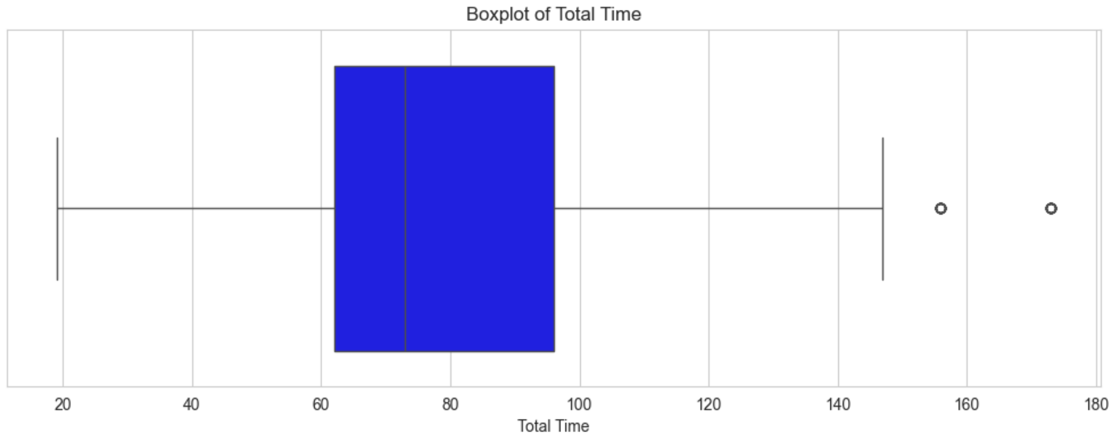
[13]:
```
sns.set_style('whitegrid')
sns.boxplot(data=df,x='post_time')
plt.title('Distribution of difference of Wheels Out vs End Time of OR')
plt.show()
```

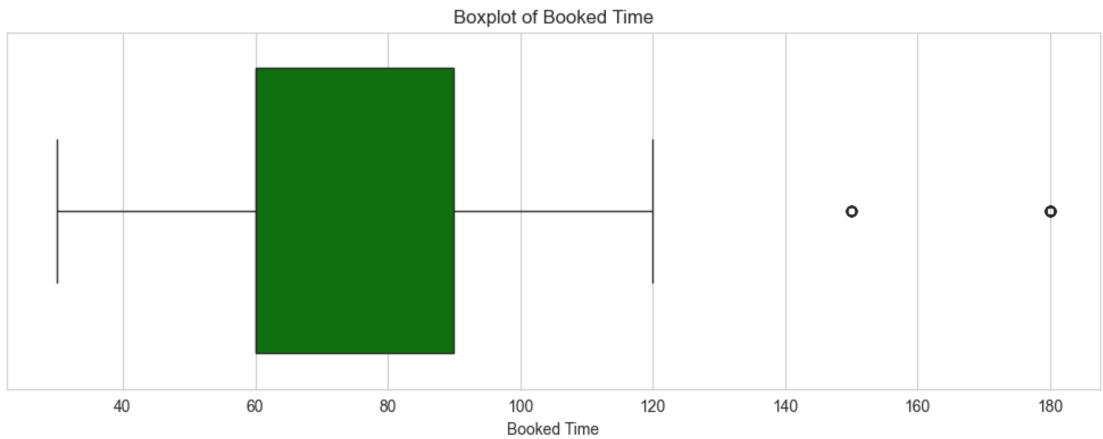


[14]:
```
sns.set_style('whitegrid')
sns.boxplot(data=df,x='diff')
plt.title('Distribution of difference of Total Time vs Booked Time of OR')
plt.show()
```



[15]:
```
fig, axes = plt.subplots(2, 1, figsize=(10, 8))

sns.boxplot(x='total_time',data=df, ax=axes[0],color='blue')
axes[0].set_title('Boxplot of Total Time')
axes[0].set_xlabel('Total Time')

sns.boxplot(x='Booked Time',data=df, ax=axes[1],color='green')
axes[1].set_title('Boxplot of Booked Time')
axes[1].set_xlabel('Booked Time')

plt.tight_layout()


plt.show()
```





0.10 Here's the description for each column with the median, mean, and standard deviation:

- **Booked Time**: Median: 60.00 mins, Mean: 77.19 mins, Std: 30.43 mins
- **Start Delay**: Median: 48.00 mins, Mean: 57.07 mins, Std: 40.60 mins
- **Preoperative Time (pre_time)**: Median: 23.00 mins, Mean: 21.53 mins, Std: 6.42 mins
- **OR Time (OR_time)**: Median: 35.00 mins, Mean: 45.48 mins, Std: 26.74 mins
- **Postoperative Time (post_time)**: Median: 13.00 mins, Mean: 12.69 mins, Std: 2.67 mins
- **Total Time (total_time)**: Median: 73.00 mins, Mean: 79.70 mins, Std: 31.82 mins
- **Difference (diff)**: Median: 3.00 mins, Mean: 2.51 mins, Std: 15.36 mins

1 Q) Is the average `Booked Time` greater than average `OR_time`?

[16]:
```
df[['Booked Time','OR_time']].describe()
```

[16]:
```
       Booked Time      OR_time
count  2172.000000  2172.000000
```

mean	77.189227	45.475138
std	30.430015	26.742297
min	30.000000	12.000000
25%	60.000000	28.000000
50%	60.000000	35.000000
75%	90.000000	58.000000
max	180.000000	136.000000

$H_0 : \mu_b <= \mu_o$

$H_1 : \mu_b > \mu_o$

- μ_b is mean `Booked time` and μ_o is mean `OR_time`.

[17]:
```
b,o=df['Booked Time'],df['OR_time']
```

1.0.1 Check for equal variance

[18]:
```
levene(b,o)
```

[18]: `LeveneResult(statistic=6.202423371947073, pvalue=0.012794650182384204)`

1.0.2 Unequal variances as pvalue of Levene test < .05

[19]:
```
ttest_ind(b,o,alternative='greater',equal_var=False)
```

[19]:
```
TtestResult(statistic=36.48461924887285, pvalue=3.0324012009754633e-254,
df=4271.501772171244)
```

1.0.3 As pvalue<.05 so we reject null. We conclude that average `Booked Time` is significantly greater than average `OR_time`.

1.0.4 The actual mean initial booked time is significantly greater than the mean surgery time which is expected as it is part of basic Healthcare practice.

2 Q) Is the average `total_time` greater than average `Booked Time`?

[20]:
```
df[['total_time','Booked Time']].describe()
```

[20]:

	total_time	Booked Time
count	2172.000000	2172.000000
mean	79.697053	77.189227
std	31.822390	30.430015
min	19.000000	30.000000
25%	62.000000	60.000000
50%	73.000000	60.000000
75%	96.000000	90.000000
max	173.000000	180.000000

$H_0 : \mu_a <= \mu_b$

$H_1 : \mu_a > \mu_b$

- μ_a is mean actual `total_time` and μ_b is mean `Booked Time (min)`.

[21]:
```
a,b=df['total_time'],df['Booked Time']
```

2.0.1 Check for equal variance

[22]:
```
levene(a,b)
```

[22]:
```
LeveneResult(statistic=7.724637519841335, pvalue=0.005470668221704837)
```

2.0.2 Unequal variances as pvalue of Levene test < .05

[23]:
```
ttest_ind(a,b,alternative='greater',equal_var=False)
```

[23]:
```
TtestResult(statistic=2.6544693918029436, pvalue=0.003986179256664374,
df=4333.337401233744)
```

2.0.3 As pvalue<.05 so we reject null. We conclude that average `total_time` is significantly greater than average `Booked Time`.

2.0.4 The mean total time spent in the OR exceeds the time initially booked, indicating inefficiencies or underestimation in the booking process.

2.1 Summation of `total_time`,`Booked Time`,`diff` over weeks.

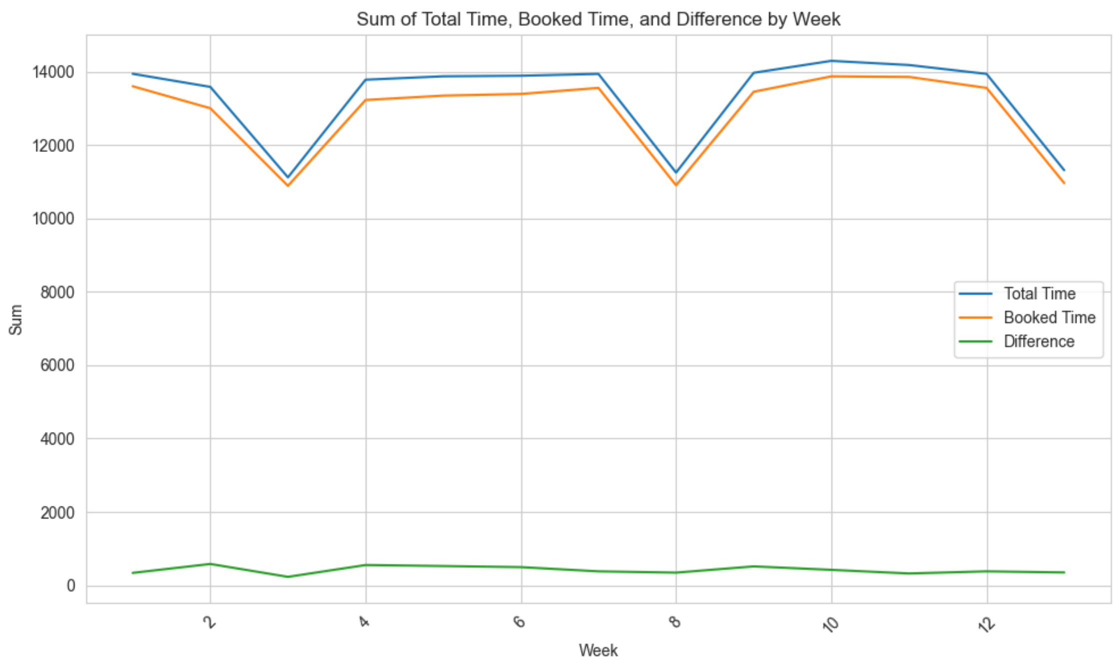
[24]:
```
grouped_df = df.groupby('week')[['total_time', 'Booked Time', 'diff']].sum().
 ↪reset_index()
grouped_df
```

[24]:

	week	total_time	Booked Time	diff
0	1	13944.0	13605	339.0
1	2	13587.0	13005	582.0
2	3	11121.0	10890	231.0
3	4	13783.0	13230	553.0
4	5	13876.0	13350	526.0
5	6	13891.0	13395	496.0
6	7	13941.0	13560	381.0
7	8	11251.0	10905	346.0
8	9	13971.0	13455	516.0
9	10	14297.0	13875	422.0
10	11	14183.0	13860	323.0
11	12	13941.0	13560	381.0
12	13	11316.0	10965	351.0

[25]:
```
plt.figure(figsize=(10, 6))
sns.lineplot(x='week', y='total_time', data=grouped_df, label='Total Time')
sns.lineplot(x='week', y='Booked Time', data=grouped_df, label='Booked Time')
```

```
sns.lineplot(x='week', y='diff', data=grouped_df, label='Difference')

plt.title('Sum of Total Time, Booked Time, and Difference by Week')
plt.xlabel('Week')
plt.ylabel('Sum')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)

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

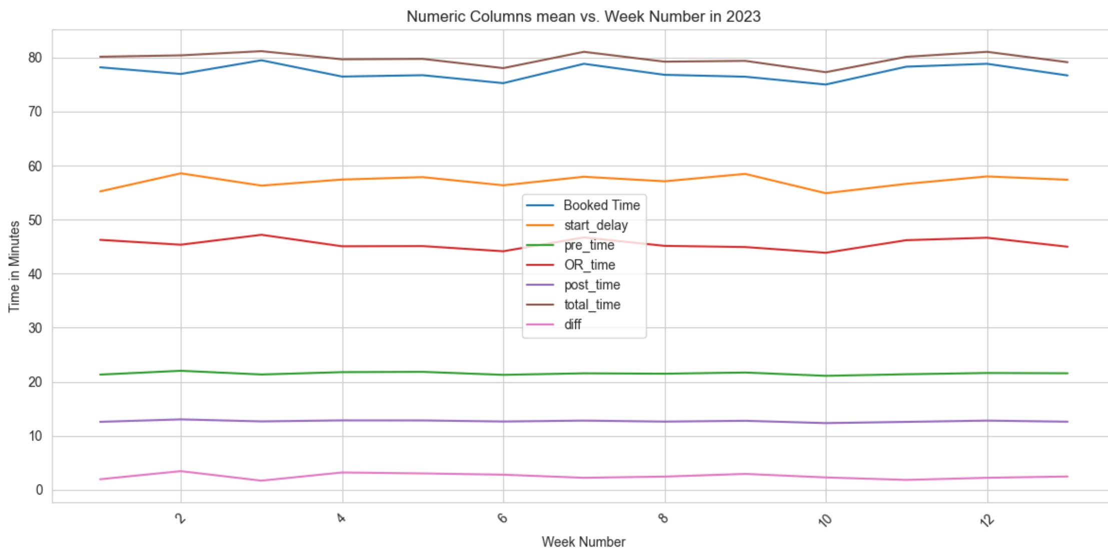


2.1.1 Sum of total_time and Booked Time exhibit periodic fluctuations but is mostly stable over weeks, indicating consistent surgical activity. Monitoring trends and adapting scheduling strategies can optimize resource utilization and improve patient care delivery.

3 Q) Is there any trends related to the mean of numeric column with respect to time?

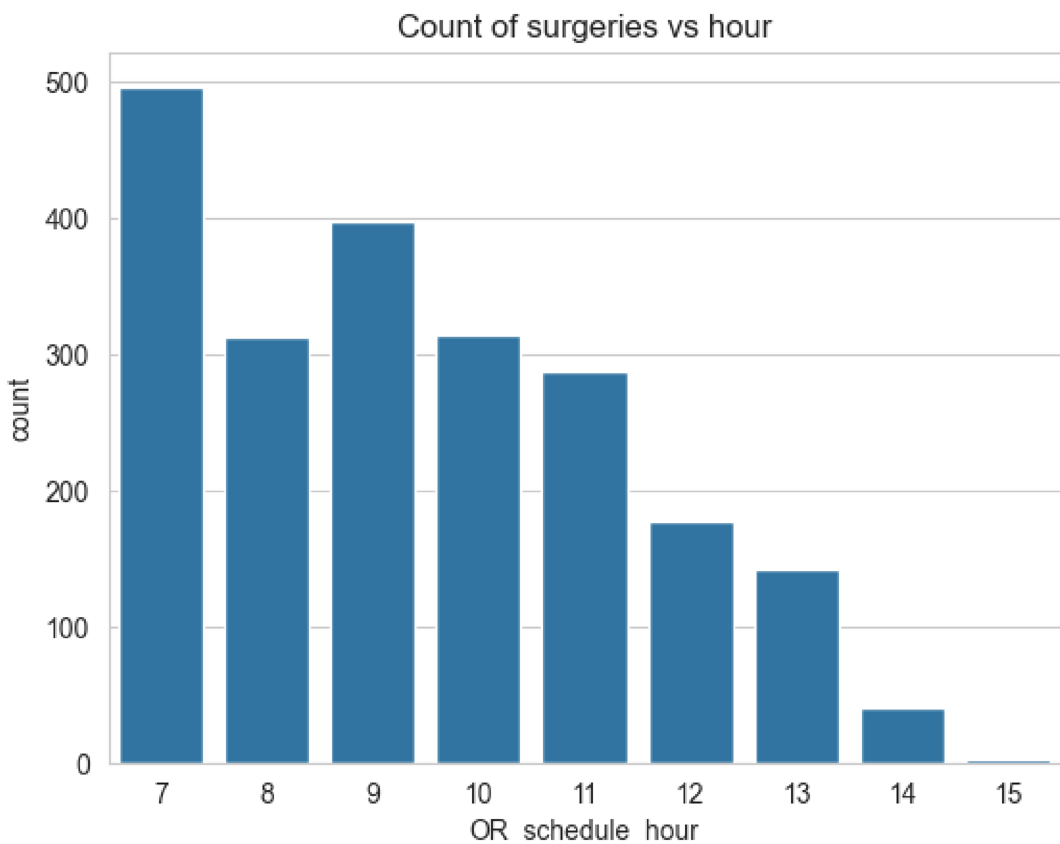
[26]:
```
numerical.pop()
numerical.pop()
weekly_means = df.groupby('week')[numerical].mean().reset_index()
plt.figure(figsize=(12, 6))
```

```
for column in numerical:
    plt.plot(weekly_means['week'], weekly_means[column], label=column)

plt.xlabel('Week Number')
plt.ylabel('Time in Minutes')
plt.title('Numeric Columns mean vs. Week Number in 2023')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)

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



3.0.1 There is no clear trend or temporal pattern.

[27]:
```
sns.countplot(data=df,x='OR_schedule_hour')
plt.title('Count of surgeries vs hour')
plt.show()
```

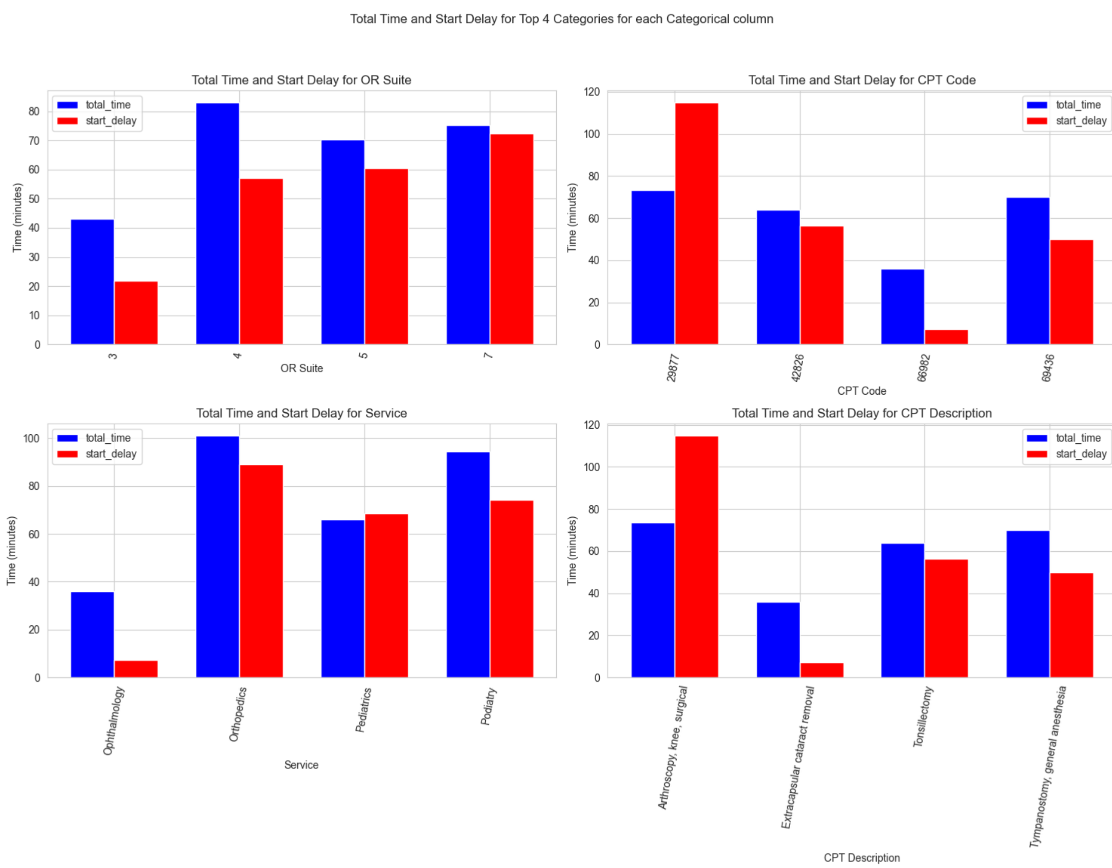


3.0.2 Most surgeries are performed in the morning during 7-11 AM.Can try to extend surgery hours to reduce cancellations and make resource utilization smoother.

3.1 Total time and start delays for common surgeries

[28]:
```
fig, axes = plt.subplots(2, 2, figsize=(15, 12))
fig.suptitle('Total Time and Start Delay for Top 4 Categories for each␣
↪Categorical column')

axes = axes.flatten()

for i, col in enumerate(categorical_columns):

    grouped_df = df.groupby(col)[['total_time', 'start_delay']]
    grouped_df=grouped_df.mean().reset_index()

    top_4_categories = df[col].value_counts().nlargest(4).index

    top_4_df = grouped_df[grouped_df[col].isin(top_4_categories)]
```

```
    bar_width = 0.35
    x = np.arange(len(top_4_df[col]))

    axes[i].bar(x - bar_width/2, top_4_df['total_time'], bar_width, label='total_time', color='b')
    axes[i].bar(x + bar_width/2, top_4_df['start_delay'], bar_width, label='start_delay', color='r')

    axes[i].set_title(f'Total Time and Start Delay for {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Time (minutes)')
    axes[i].set_xticks(x)
    axes[i].set_xticklabels(top_4_df[col], rotation=80)
    axes[i].legend()
    axes[i].grid(True)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



3.1.1 Analyzing historical `total_time` and `start_delay` data allows for more accurate scheduling of common surgeries, reducing delays and improving OR efficiency. This data-driven approach ensures better alignment of scheduled times with actual durations, optimizing resource allocation and minimizing downtime.

3.2 Analyzing the Utlization rates

We'll analyze

- Overall Utilization rates
- Monthly Utlization rates
- Utilization rates for each week
- Utilization rate by OR suite

3.2.1 Overall utlization rate

[29]:
```
total_days = df['Date'].nunique()
total_or_suites = df['OR Suite'].nunique()
total_available_time = total_days * total_or_suites * 10 * 60  # 10 hours/day *␣
↪60 minutes/hour
total_actual_or_time = df['total_time'].sum()
overall_utilization_rate = (total_actual_or_time / total_available_time) * 100
overall_utilization_rate = round(overall_utilization_rate, 2)
print(f'Overall Utilization Rate: {overall_utilization_rate}%')
```

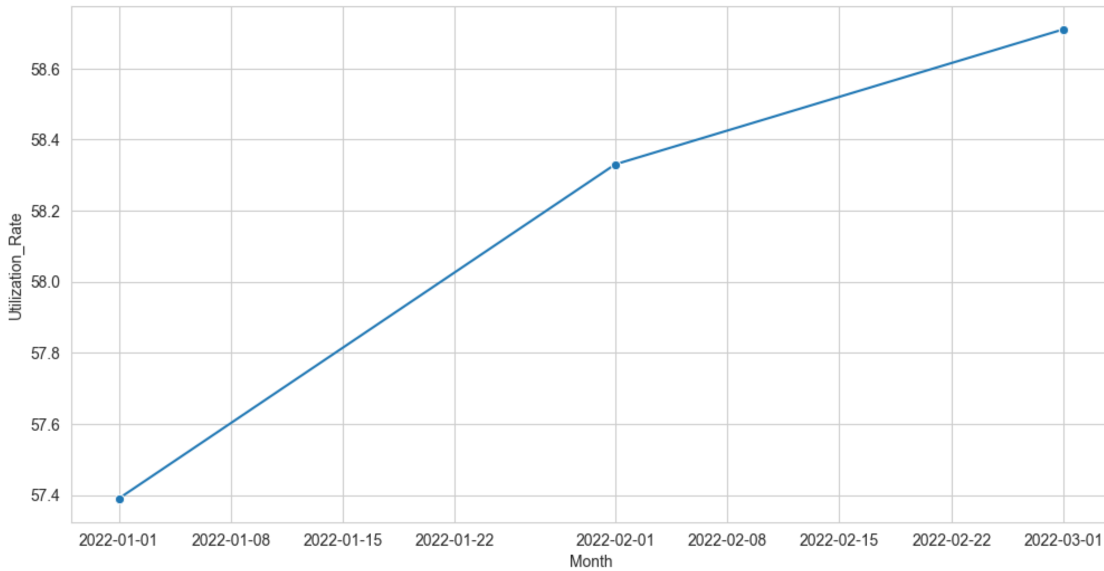
```
Overall Utilization Rate: 58.17%
```

3.2.2 Monthly utlization rate

[30]:
```
df['Month'] = df['Date'].dt.to_period('M').astype(str)
workdays_per_month = df.groupby('Month')['Date'].nunique().
↪reset_index(name='Workdays')
workdays_per_month['Total_Available_Time'] = workdays_per_month['Workdays'] *␣
↪10 * 60 * df['OR Suite'].nunique()
monthly_actual_or_time = df.groupby('Month')['total_time'].sum().reset_index()
monthly_utilization = pd.merge(workdays_per_month, monthly_actual_or_time,␣
↪on='Month')
monthly_utilization['Utilization_Rate'] = (monthly_utilization['total_time'] /␣
↪monthly_utilization['Total_Available_Time']) * 100
monthly_utilization['Utilization_Rate'] =␣
↪monthly_utilization['Utilization_Rate'].round(2)

print(monthly_utilization)
```

```
     Month  Workdays  Total_Available_Time  total_time  Utilization_Rate
0  2022-01        20                 96000     55092.0             57.39
1  2022-02        19                 91200     53199.0             58.33
2  2022-03        23                110400     64811.0             58.71
```

[31]:
```
monthly_utilization['Month'] = pd.to_datetime(monthly_utilization['Month'])
plt.figure(figsize=(12, 6))
sns.lineplot(x='Month', y='Utilization_Rate', data=monthly_utilization,␣
  ↪marker='o')
plt.show()
```



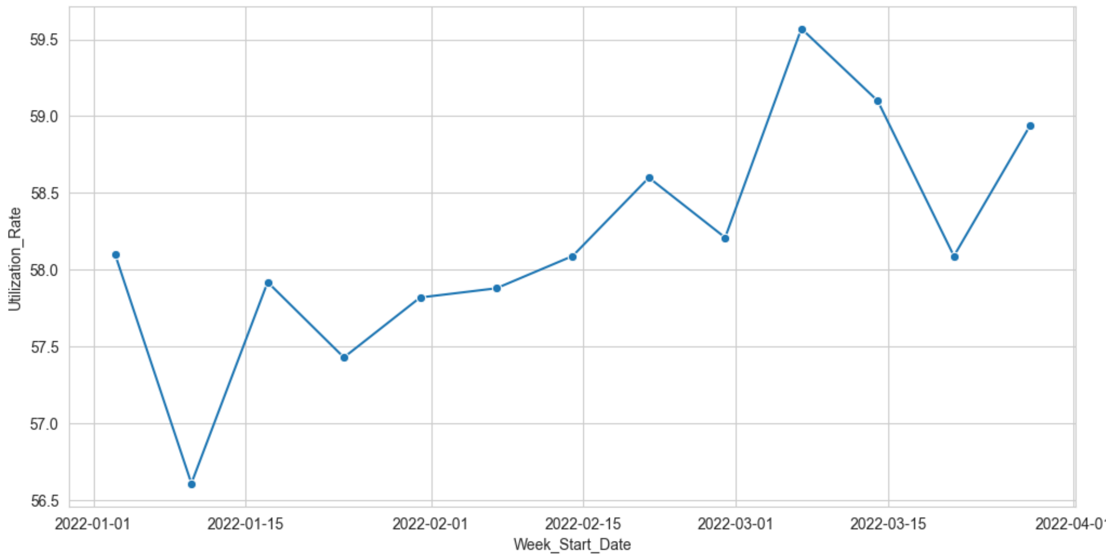
3.3 If we look at the month-wise utlization rates the OR's are running at roughly 60% of their capcity and March Month saw highest utlization of the OR

Weekly utlization rate

[32]:
```
df['Week'] = df['Date'].dt.to_period('W').astype(str)
workdays_per_week = df.groupby('Week')['Date'].nunique().
  ↪reset_index(name='Workdays')
workdays_per_week['Total_Available_Time'] = workdays_per_week['Workdays'] * 10␣
  ↪* 60 * df['OR Suite'].nunique()
weekly_actual_or_time = df.groupby('Week')['total_time'].sum().reset_index()
weekly_utilization = pd.merge(workdays_per_week, weekly_actual_or_time,␣
  ↪on='Week')
weekly_utilization['Utilization_Rate'] = (weekly_utilization['total_time'] /␣
  ↪weekly_utilization['Total_Available_Time']) * 100
weekly_utilization['Utilization_Rate'] = weekly_utilization['Utilization_Rate'].
  ↪round(2)
print(weekly_utilization.head())
weekly_utilization['Week_Start_Date'] = weekly_utilization['Week'].apply(lambda␣
  ↪x: x.split('/')[0])
```

```python
weekly_utilization['Week_Start_Date'] = pd.
    ↪to_datetime(weekly_utilization['Week_Start_Date'])
print(weekly_utilization.head())
plt.figure(figsize=(12, 6))
sns.lineplot(x='Week_Start_Date', y='Utilization_Rate',␣
    ↪data=weekly_utilization, marker='o')
plt.show()
```

```
                    Week  Workdays  Total_Available_Time  total_time  \
0  2022-01-03/2022-01-09         5                 24000     13944.0
1  2022-01-10/2022-01-16         5                 24000     13587.0
2  2022-01-17/2022-01-23         4                 19200     11121.0
3  2022-01-24/2022-01-30         5                 24000     13783.0
4  2022-01-31/2022-02-06         5                 24000     13876.0

   Utilization_Rate
0             58.10
1             56.61
2             57.92
3             57.43
4             57.82
                    Week  Workdays  Total_Available_Time  total_time  \
0  2022-01-03/2022-01-09         5                 24000     13944.0
1  2022-01-10/2022-01-16         5                 24000     13587.0
2  2022-01-17/2022-01-23         4                 19200     11121.0
3  2022-01-24/2022-01-30         5                 24000     13783.0
4  2022-01-31/2022-02-06         5                 24000     13876.0

   Utilization_Rate Week_Start_Date
0             58.10      2022-01-03
1             56.61      2022-01-10
2             57.92      2022-01-17
3             57.43      2022-01-24
4             57.82      2022-01-31
```



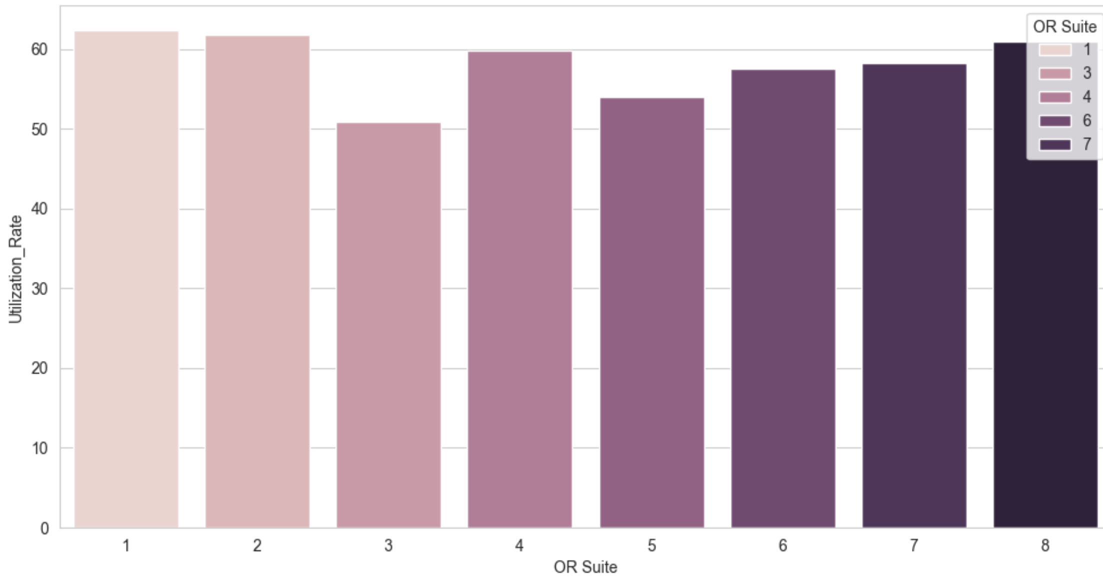
- We can see that 2nd week of Januaray month had the lowest utlization of the OR rooms whereas in 2nd week of march saw the highest utlization of OR.
- After January month the utlization has been on the higher side

Utlization by OR Suite

[33]:
```
workdays_per_suite = df.groupby('OR Suite')['Date'].nunique().
  ↪reset_index(name='Workdays')
workdays_per_suite['Total_Available_Time'] = workdays_per_suite['Workdays'] *␣
  ↪10 * 60
actual_or_time_per_suite = df.groupby('OR Suite')['total_time'].sum().
  ↪reset_index()
suite_utilization = pd.merge(workdays_per_suite, actual_or_time_per_suite,␣
  ↪on='OR Suite')
suite_utilization['Utilization_Rate'] = (suite_utilization['total_time'] /␣
  ↪suite_utilization['Total_Available_Time']) * 100
suite_utilization['Utilization_Rate'] = suite_utilization['Utilization_Rate'].
  ↪round(2)
print(suite_utilization)
plt.figure(figsize=(12, 6))
sns.barplot(x='OR Suite', y='Utilization_Rate', data=suite_utilization, hue='OR␣
  ↪Suite')
plt.show()
```

```
   OR Suite  Workdays  Total_Available_Time  total_time  Utilization_Rate
0         1        62                 37200     23205.0             62.38
1         2        62                 37200     22955.0             61.71
2         3        62                 37200     18911.0             50.84
3         4        62                 37200     22215.0             59.72
```

4	5	62	37200	20100.0	54.03
5	6	62	37200	21408.0	57.55
6	7	62	37200	21634.0	58.16
7	8	62	37200	22674.0	60.95



3.3.1 We can see that OR Suite 1,2 and 8 have a high utlization rate and OR suite 3 has the lowest utlization rate desipte of the fact that most of the operations are perfomred in this OR Suite.

3.4 Recommendations to Improve OR Efficiency

Reference for different workflow time components

- **Booked Time**: The scheduled duration for the use of OR.
- **start_delay**: The delay between the scheduled start time and the actual start time.
- **pre_time**: Time spent on preoperative procedures.
- **OR_time**: Actual time spent for surgery.
- **post_time**: Time spent on postoperative procedures.
- **total_time**: The total time from the start to the end of the entire surgical process.
- **diff**: The difference of total time and booked time.

To influence `pre_time`, `OR_time`, and `post_time`, we need better trained staff and doctors.

Other than that, the major issues are the `start_delay` and the difference between actual `total_time` and `Booked Time`.

3.4.1 Recommendations to Overcome Major Issues

Issue 1: Start Delay (Median: 48 minutes)

1. **Streamline Preoperative Processes**: Implement checklists and pre-op coordination to reduce delays.

2. **Improve Patient Preparation**: Ensure patients are ready on time by enhancing communication and preparation protocols.

3. **Optimize Staff Schedules**: Align staff schedules to minimize waiting times and ensure timely availability of surgical teams.

4. **Utilize Time After 11AM**: Extend operating hours after 11 AM for smoother resource utilization, balancing the allocation of time and reducing expenses.

5. **How Much Resource Can be Saved**: This data is for 3 months so if we save 15 mins on average we will be saving 15*4*2172 minutes which is equal to 1785 more surgeries. Estimated annual extra profit will be ((1785/(4*2172))*100 => 20%) above 10% even if we subtract staff, surgery and resource cost.

Issue 2: Total Time Exceeds Booked Time

1. **Refine Booking Estimates**: Use historical data to provide more accurate booking times and start delays for different types of surgeries.

2. **Further Segment the Procedures**: Collect more data and further segment the stages in a surgery to pinpoint the stage where the efficiency can be further improved.

3. **Utilize OR Suites uniformly**: OR Suite 3 is most frequently used, can try to evenly share the load among other suites to ensure uniform resource utilization.

Issue 3: OR Utilization

1. **Overall, monthly, weekly rates**: Utilization is around 60% assuming 10 hours availability for each day.Rates are mostly higher towards March. Utilization can be increased with proper guidelines and without reducing treatment quality.

2. **OR Suite wise rate**: Average rate is around 60% with OR Suite 3 having a bit less value despite being the most frequently used OR suite. Utilization can be increased with proper guidelines and without reducing treatment quality.

```
[ ]:
```