Call Volume Trend Analysis

FINAL PROJECT-4

Project By:

Yaadhav R

Project Description

This project delves into Customer Experience (CX) analytics, focusing on the inbound calling team of a company. Spanning 23 days, the dataset provides details like agent names, queue times, call timestamps, durations, and call statuses (abandoned, answered, or transferred). The CX team, pivotal in any organization, analyzes customer feedback and data to drive insights for the organization. With AI-powered tools like IVR, RPA, Predictive Analytics, and Intelligent Routing, the project aims to optimize inbound customer support. By handling incoming calls effectively, the goal is to attract, engage, and delight customers, fostering long-term loyalty and advocacy.

Tech-Stack Used

Jupyter Notebook - Version 7.0.7 is used to implement Python with various other libraries :

- Pandas - Data Manipulation and Analysis.
- Numpy - Mathematical Computations.
- Matplotlib - Data Visualization.
- Seaborn - Advanced Data Visualization.

Microsoft Excel 2023 - Version 16.80 is used in this project as it is a simple and easy to use software.

Tasks

- A. Average Call Duration
- B. Call Volume Analysis
- C. Manpower Planning
- D. Night Shift Manpower Planning

Average Call Duration

Approach

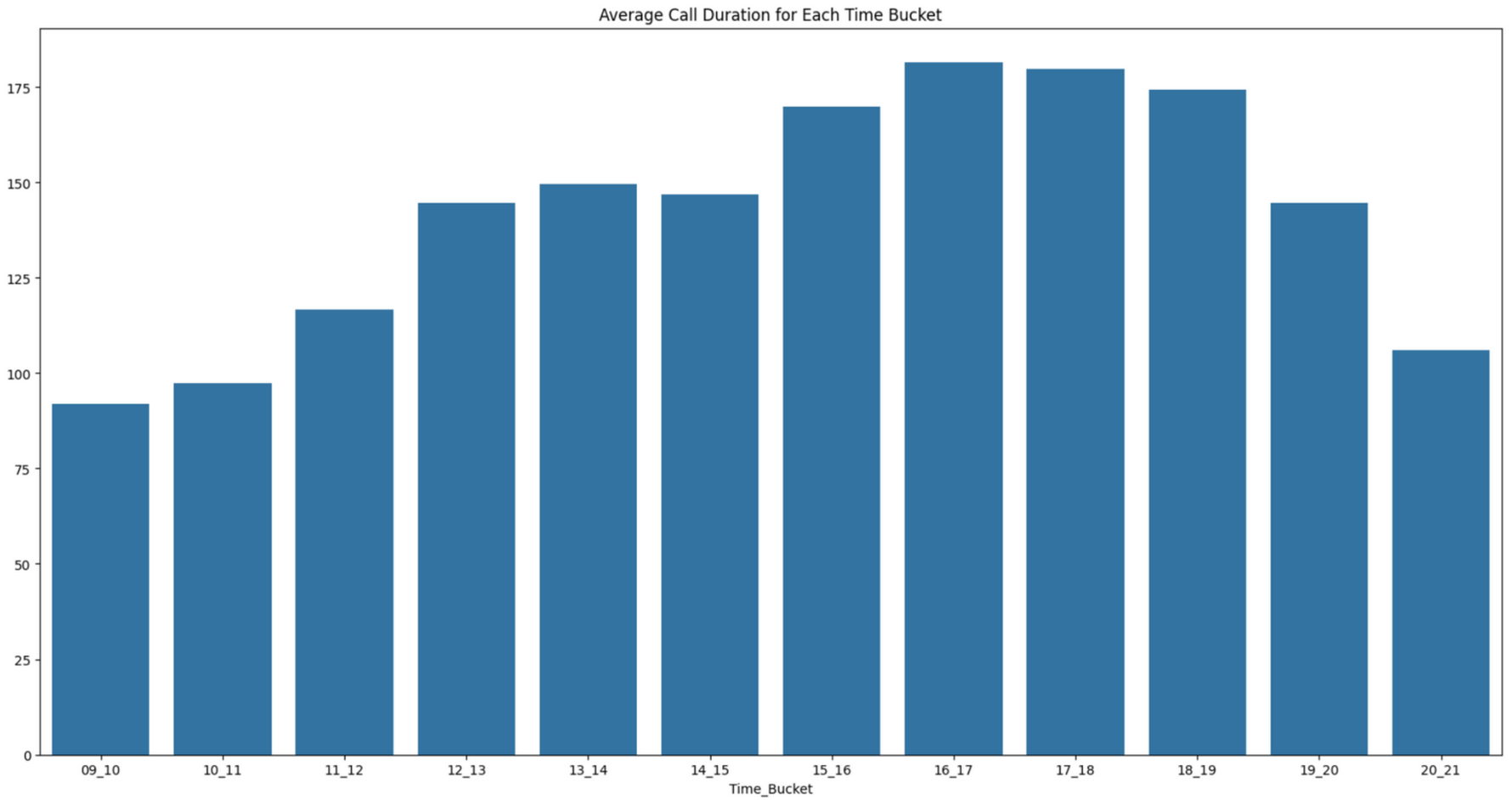
- The ‘data’ dataframe is grouped by ‘Time_Bucket’ .
- Now the average of the ‘Call_Duration’ is calculated on this grouped data.
- The result is stored in avg_time and barplot() is used on it as shown.

```
gd = data.groupby(cols[6])
avg_time = gd[cols[8]].mean()
```

```
plt.figure( figsize=(20,10) )

sns.barplot(x=avg_time.index, y=avg_time.values)
plt.title("Average Call Duration for Each Time Bucket")

plt.show()
```

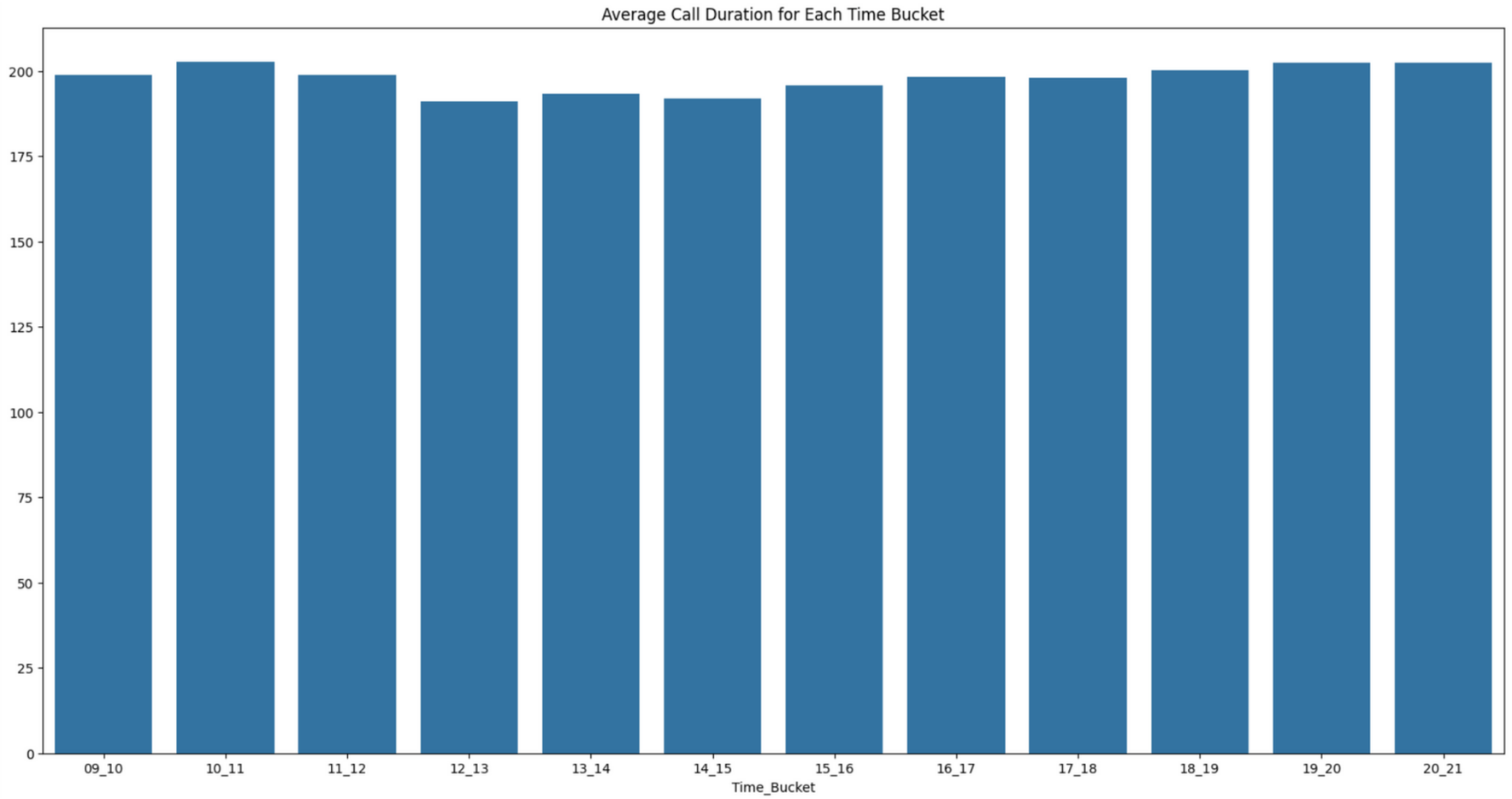
- The same steps are repeated on the data excluding calls that are abandon with call duration of 0s.

Average Call Duration - Including Abandon Calls



Time_Bucket	Call_Seconds (s)
09_10	92.0
10_11	97.0
11_12	117.0
12_13	145.0
13_14	150.0
14_15	147.0
15_16	170.0
16_17	181.0
17_18	180.0
18_19	174.0
19_20	145.0
20_21	106.0

Average Call Duration - Excluding Abandon Calls



Time_Bucket	Call_Seconds (s)
09_10	199.0
10_11	203.0
11_12	199.0
12_13	191.0
13_14	193.0
14_15	192.0
15_16	196.0
16_17	198.0
17_18	198.0
18_19	200.0
19_20	202.0
20_21	203.0

Insights

- The average call duration exhibited a trend of increase until the 16_17 time bucket, reaching its peak with 181 s. However, it began to decrease afterwards.
- Upon excluding abandoned calls, the average call duration showed a narrower range, with the highest observed in the 16_17 time bucket at 203 s and the lowest in the 09_10 time bucket at 191 s.
- The fluctuations in average call duration could be attributed to abandoned calls skewing the data.
- Excluding abandoned calls provides a clearer picture of actual call handling efficiency, revealing a more uniform distribution of average durations across time buckets.

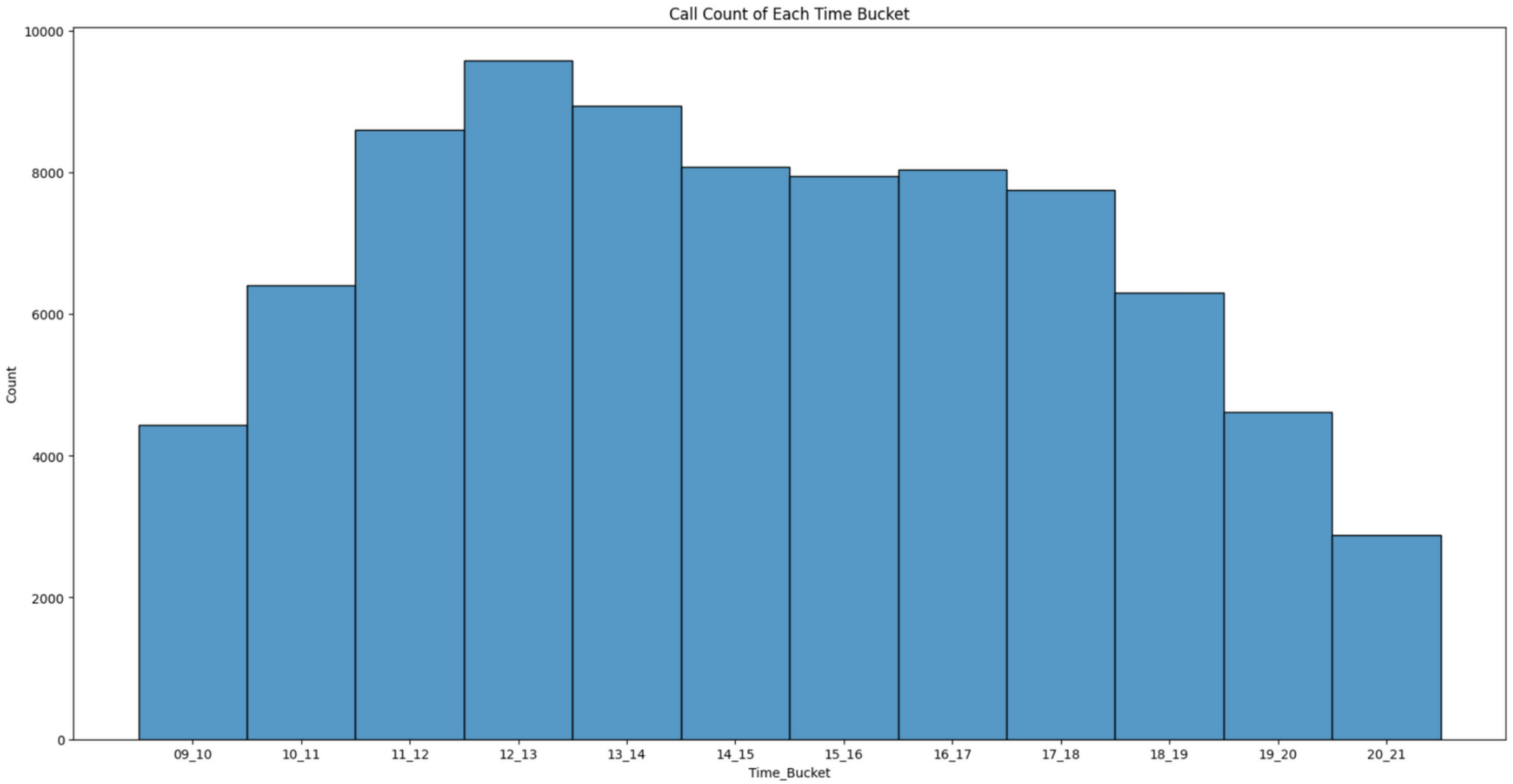
Call Volume Analysis

Approach

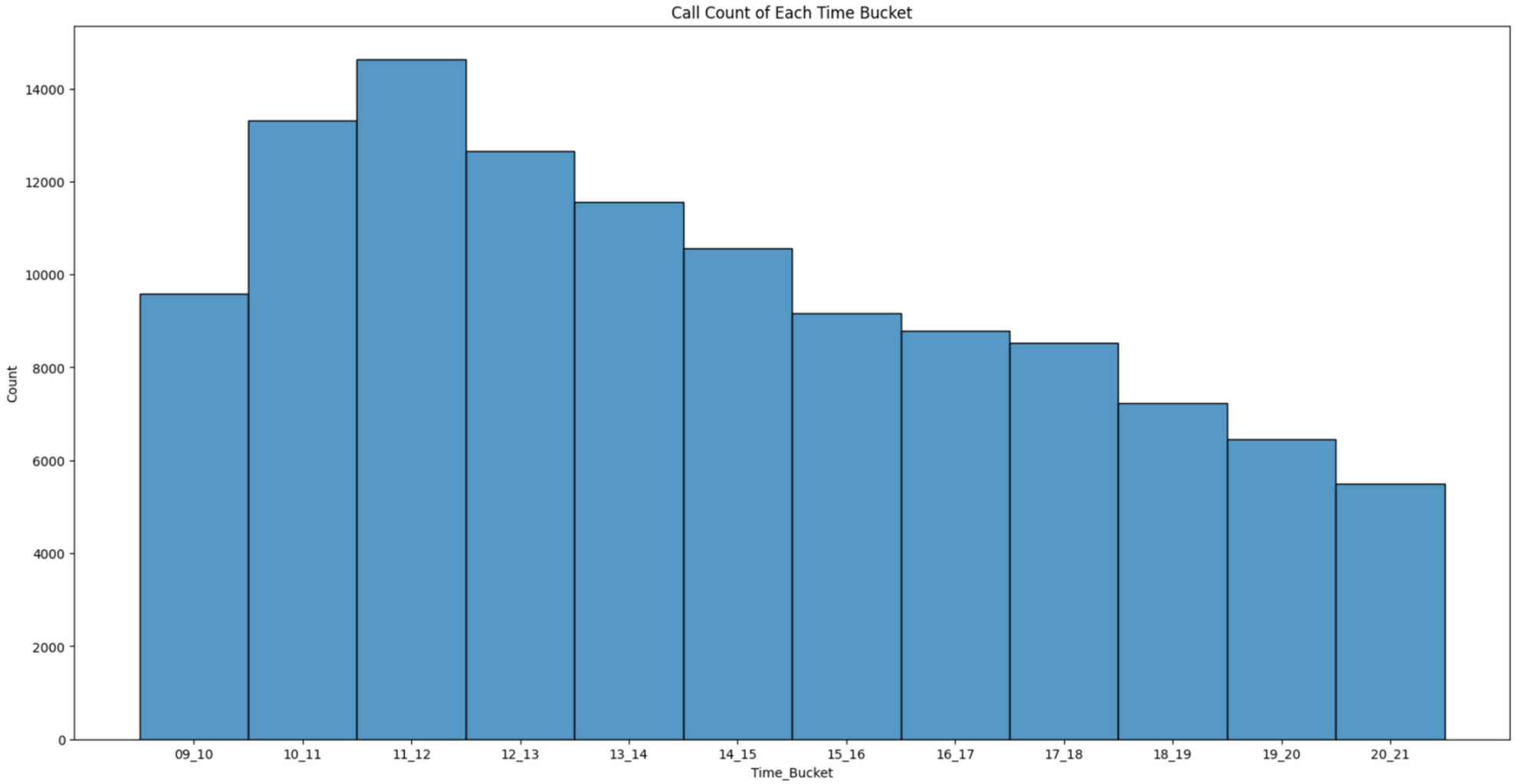
- The data's 'Time_Bucket' column is passed to the sns.histplot() function.
- This generates a histogram based on the values in the 'Time_Bucket' column.
- data2 is created by filtering out the rows with abandon calls.
- And now, data2's ‘Time_Bucket’ column is passed onto sns.histplot().

```
plt.figure(figsize=(20,10))

sns.histplot(data[cols[6]])
plt.title("Call Count of Each Time Bucket")

plt.show()
```

Call Volume - Including Abandon Calls



Call Volume - Excluding Abandon Calls



Insights

- In the case the Call Volume analysis, it observed that the trend was similar before and after removing abandoned calls.
- Including abandoned calls in the analysis revealed the highest call count in the time bucket 12_13, with a consistent increase followed by a decrease in volume.
- Excluding abandoned calls shifted the peak to an earlier time bucket, 11_12, and the subsequent decrease in call volume began earlier.
- The significant difference in the pattern occurred in the time bucket 12_13, which registered the highest call volume when abandoned calls were included, but did not retain its highest position after the removal of abandoned calls..

Manpower Planning

Approach

- First the percentage of calls answered, abandon and removed are calculated in each Time_Bucket and stored in ‘bs’.
- Then only the percentage of answered calls are filtered out from bs, which is stored in ‘ans_rate’.
- Now, number of agents working in each Time_Bucket is stored in ‘agents’.
- The ‘ans_rate’ is divided by ‘agents’, and multiplied by 90 to get the number of agents required to increase the answer rate to 90.
- This is stored in ‘min_req’ and barplot() is used to visualize.

```
tot = gd[cols[9]].count()
bs = (gd[cols[9]].value_counts()/tot)*100
```

```
bs = bs.to_frame().reset_index()
bs.rename( columns = {0:'Rate'}, inplace=True)
```

```
bs
```

```
ans_calls = bs[bs['Call_Status']=='answered']
ans_calls['Rate']
```

```
ans_rate = pd.Series( ans_calls['Rate'].values, index=ans_calls['Time_Bucket'] )
ans_rate
```

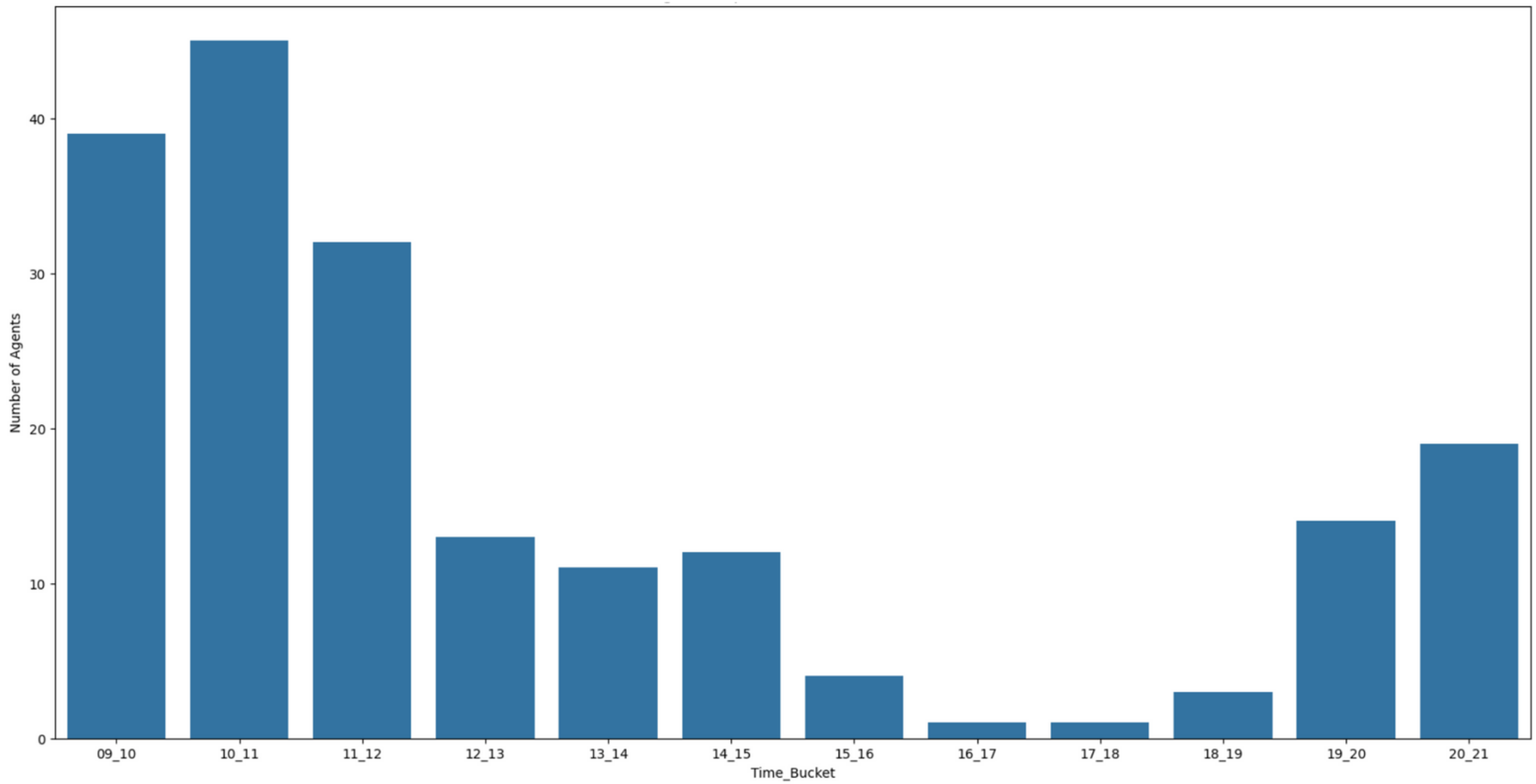
```
agents = gd[cols[1]].nunique()
agents
```

```
req = np.ceil((agents/ans_rate)*90)
req
```

```
min_req = (req-agents).astype(int)
min_req
```

```
plt.figure(figsize=(20,10))

sns.barplot( x=min_req.index, y=min_req.values)
plt.title("Agents required for Night shift")
plt.xlabel('Time_Bucket')
plt.ylabel('Number of Agents')

plt.show()
```

Agents Required to Reduce the Abandon Rate to 10%



Time_Bucket	
09_10	39
10_11	45
11_12	32
12_13	13
13_14	11
14_15	12
15_16	4
16_17	1
17_18	1
18_19	3
19_20	14
20_21	19

Insights

- A total of 194 agents are required to reduce the percentage of abandoned calls from 30% to 10%.
- The distribution of these 194 agents is inconsistent across different time buckets due to the variation in the percentage of abandoned calls.
- The percentage of abandoned calls was nearly 50% for the first three time buckets, so these 3 time buckets alone require 116 agents out of the total 194 agents, which is almost 60%.
- But during time buckets 15_16 to 18_19, the percentage of abandoned calls was already close to 10%, so these 4 time buckets require only 9 agents in total, which is just 4.6%.
- The remaining time buckets require an average of 14 agents.

Night Shift Manpower Planning

Approach

- First the given distribution is stored in 'dist' Series.
- This 'dist' values are divided by 100 and multiplied total number of calls to get expected number of calls in each Time_Bucket, which stored on calls.
- Then the number of agents required to maintain abandon rate at 10% is calculated from 'calls' using the ratio of average agents to average calls which are found from the 'req' and 'gd' reespectively.
- Finally boxplot() function is used for visualization.

```
dist = pd.Series([ 3, 3, 2, 2, 1, 1, 1, 1, 3, 4, 4, 5],
                 index=[ '21_22', '22_23', '23_00', '00_01', '01_02', '02_03', '03_04', '04_05', '05_06', '06_07', '07_08', '08_09'])
```

```
calls  = np.round((dist/100)*data.shape[0])
calls
```

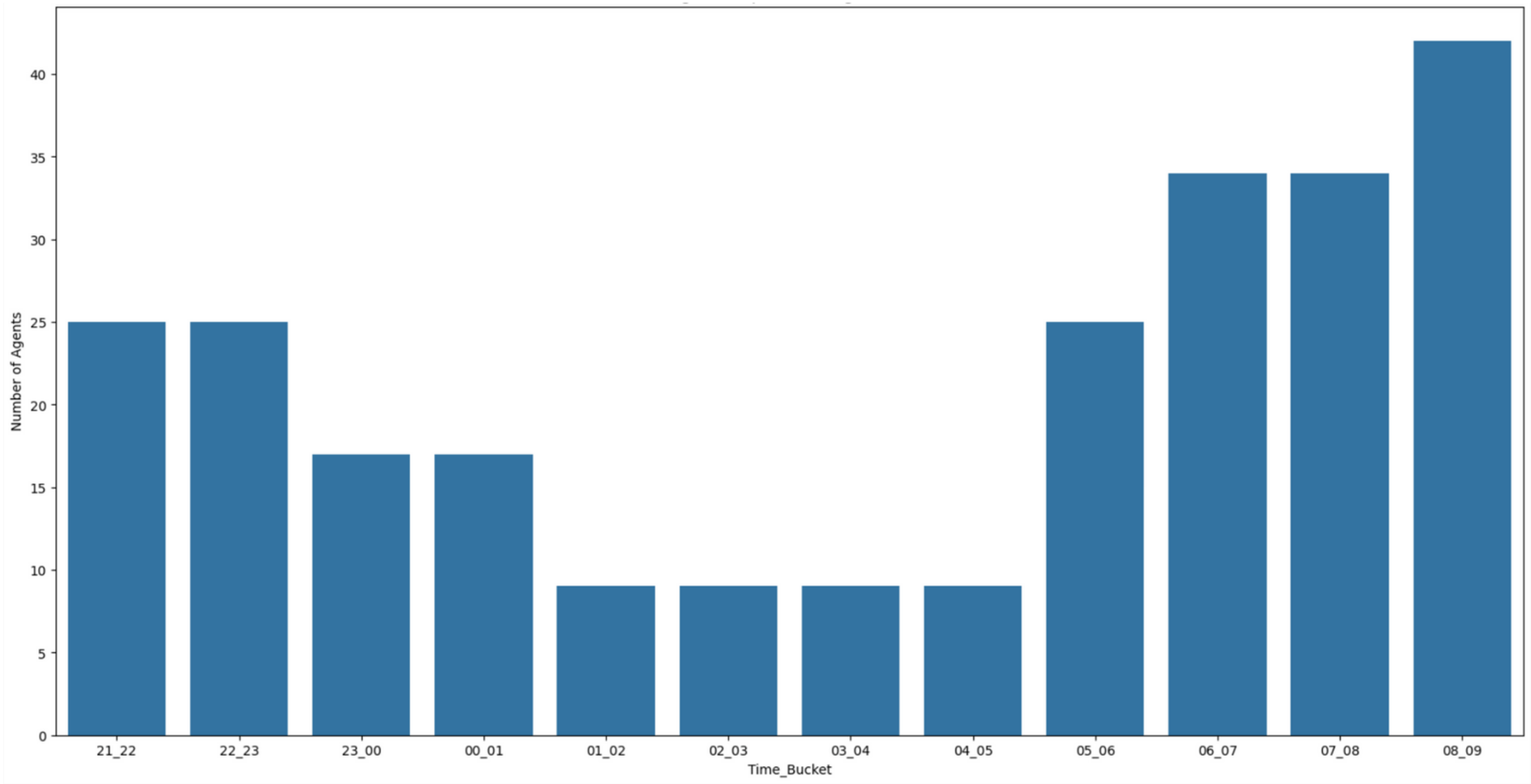
```
avg_agents = np.ceil(req.mean())
avg_agents
```

```
avg_calls = int(gd[cols[2]].count().mean())
avg_calls
```

```
agents_req = np.ceil((avg_agents/avg_calls)*calls).astype(int)
agents_req
```

```
plt.figure(figsize=(20,10))

sns.barplot( x=agents_req.index, y=agents_req.values)
plt.title("Agents required for Night shift")
plt.xlabel('Time_Bucket')
plt.ylabel('Number of Agents')

plt.show()
```

Agents Required for Night Shift



Time_Bucket	
21_22	25
22_23	25
23_00	17
00_01	17
01_02	9
02_03	9
03_04	9
04_05	9
05_06	25
06_07	34
07_08	34
08_09	42

Insights

- A total of 225 agents are required to maintain the abandoned call rate at 10% during night shifts. The distribution of these agents precisely matches the given distribution as it was derived from this.
- In the original distribution, time buckets from 01_02 to 04_05 each had 1 call, now requiring 9 agents each, representing 16%.
- Time buckets 23_00 and 00_01, which previously had 2 calls each, now require 17 agents each, accounting for 15.11%. Similarly, time buckets 21_22, 22_23, and 05_06 with 3 calls each, now require 25 agents each, representing 33.33%.
- Subsequently, time buckets 06_07 and 07_08, with 4 calls, now requires 34 agents each, making up 30.22%. And finally, time bucket 08_09, with 5 calls, now requires 42 agents, accounting for 18.66%.

Results

- First of all, this project helped to understand the concepts I learned in a better and interesting way.
- So now I feel confident in preprocessing any kind of data, as I was able to complete all the given tasks.
- It was so exciting to analyze the given data and derive insights from it.
- Overall, it was great to experience applying the skills and learn along the way.
- Looking forward to facing the upcoming challenges with confidence and excitement.

Jupyter Notebook Github Link : 🔗