**ABC TREND ANALYSIS**

**Introduction**

In this project, you'll be diving into the world of Customer Experience (CX) analytics, specifically focusing on the inbound calling team of a company. You'll be provided with a dataset that spans 23 days and includes various details such as the agent's name and ID, the queue time (how long a customer had to wait before connecting with an agent), the time of the call, the duration of the call, and the call status (whether it was abandoned, answered, or transferred).

The project circles around the data that includes the Insurance customer support call volume information. The task here is to analyze the data and draw insight to improve on our customer expeirince.

Customer experience can be the key aspect of growth for any business. A satisfied customer stays longer with the company. And is more likely to refer our business to friends and family. In the long run it also helps to build a good brand identity among the people, which in turn can boost up our insurance company growth

**Introduction about the data –**

**The data contain following Features –**

* **Agent Name –** The agent who available for handling the calls.
* **Agent ID –** Unique Id given to each and every customer support executive/agent
* **Customer Phone No –** Phone no from which we get the call
* **Queue Time –** Talks about how much time the customer has wait to talk to representative.
* **Data and time –** Data and time of the phone call.
* **Time –** Time of the phone call
* **Time Bucket –** One hour window to determine at what hourly time span we receive phone call.
* **Duration -** The duration of the phone call.
* **Call Seconds –** The call duration in second.
* **Call Status –** Call handled by executive, machine or abandoned.
* **Ringing –** Did the phone ring.

**SOFTWARE USED IN PROJECT**

To complete this project, I use the Python – 3.8.

Also here are the libraries that I use for cleaning, analysis and visualization of the data.

import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import missingno as msno

import matplotlib.pyplot as plt

**FILES USED IN THE PROJECT**

Following are the Files I used to perform this task:

**Call\_Volume\_Trend\_Analysis\_Project\_9.xlsx –** Contain the Data about call volume.

**project\_9.ipynb –** Contains the Code that I use for cleaning, analyzing , visualize the data.

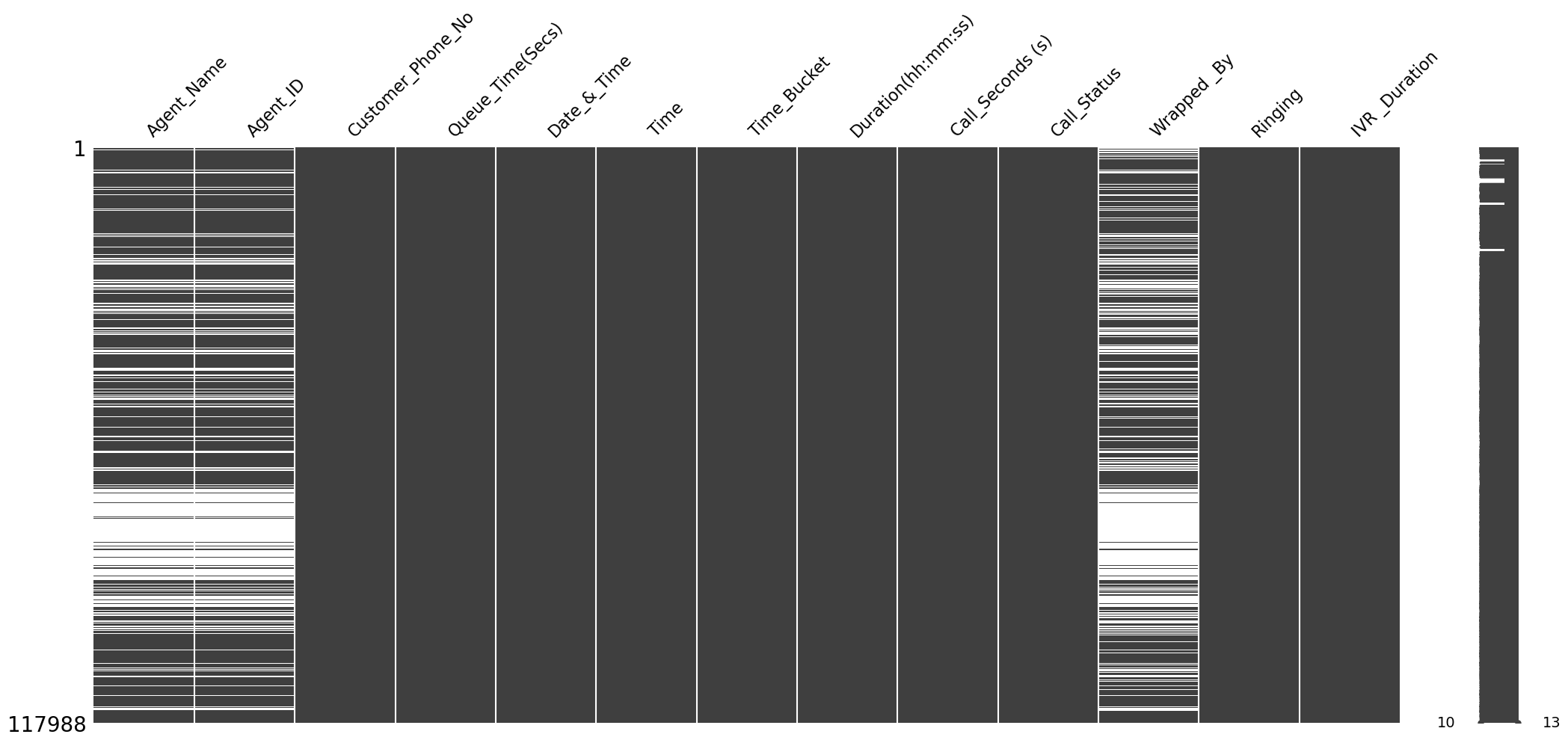
**Call\_Volume\_Trend\_Analysis.docx –** Contains detail and step by step guide about How I performs the task.

**Graphs - Contains** the images of graphs that I use for visualization.

**Sheets –** Contains the Sheets with analysis.

**ANALYSIS**

**Missing Value Analysis**



As usual process I first look at the missing value in the data.

**Agent Name 34198**

**Agent ID 34198**

**Customer Phone No 0**

**Queue Time (Secs) 0**

**Date & Time 0**

**Time 0**

**Time Bucket 0**

**Duration(hh:mm:ss) 0**

**Call\_Seconds (s) 0**

**Call\_Status 0**

**Wrapped \_By 47877**

**Ringing 0**

**IVR \_Duration 0**

**dtype: int64**

Finding there is clear pattern in the missing values of the data the calls that are abandoned have missing blanks in all the pattern. Therefore, I have performed the following steps in my analysis for the data.

**Agent Name -** I have fill the Na with the Abandon to denote there are no agents to take the call.

**Wrapped By –** I have used the term Abandoned again to denote the abandoned call.

**Agent Id –** I have used 000000 to denote the Agent that are not present and call goes the abandon.

Steps I took –

df['Agent\_Name'].fillna('Abandon', inplace=True)

df['Agent\_ID'].fillna(0000000, inplace=True)

df['Wrapped \_By'].fillna('Abandon', inplace=True)

There are no duplicates in the data so there is no need to take care of them. Apart from that the data is very well collected, neat and clean. So, no further steps have been taken to clean the data.

**EDA**

**Average Call Duration**

**Determine the average duration of all incoming calls received by agents. This should be calculated for each time bucket.**

**Your Task: What is the average duration of calls for each time bucket?**

For the Analysis of this task, I have used the following code. Also, I be submitting the excel sheet that contain the table of the analysis.

**Average Duration Per Bucket Autowrapped**

This includes the calls duration per bucket for the calls that are autowrapped or handled by machine. Here are the time taken for the each bucket as follows -

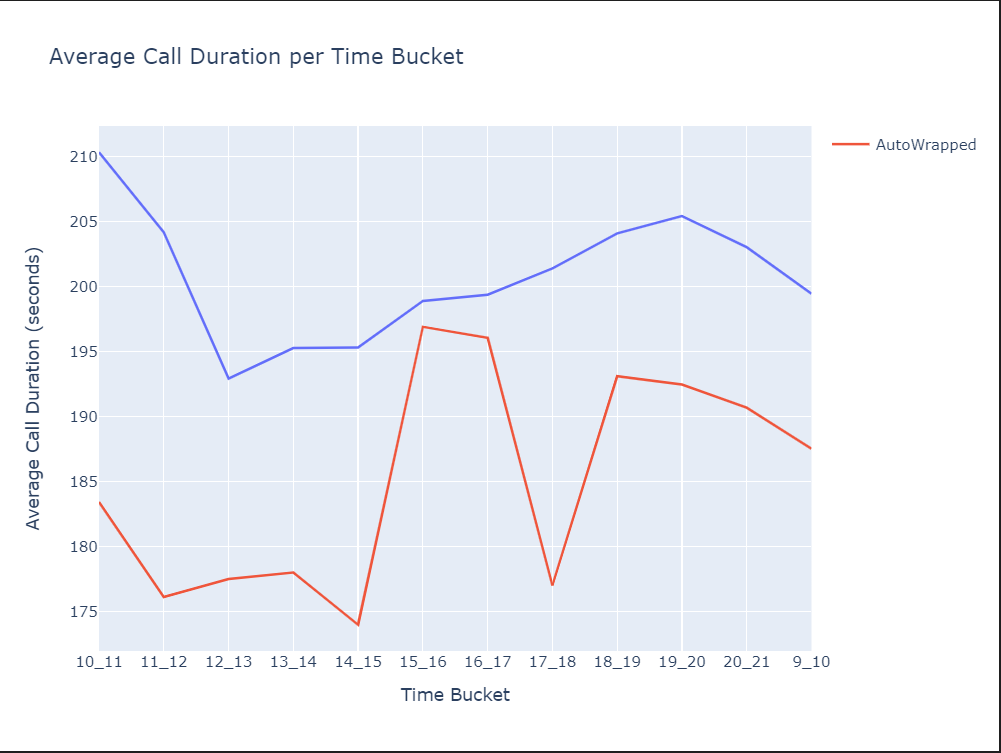
|  |  |
| --- | --- |
| **Time\_Bucket** | **Call\_Seconds (s)** |
| 10\_11 | 183.4392 |
| 11\_12 | 176.139 |
| 12\_13 | 177.5283 |
| 13\_14 | 178.013 |
| 14\_15 | 174.006 |
| 15\_16 | 196.9043 |
| 16\_17 | 196.06 |
| 17\_18 | 177.0296 |
| 18\_19 | 193.1104 |
| 19\_20 | 192.4724 |
| 20\_21 | 190.699 |
| 9\_10 | 187.5392 |
|  |  |

**Average Duration Per Bucket by Agent -**

Here is the call duration it takes in each phone bucket by customer support executive to handle or resolve the customer’s query.

|  |  |
| --- | --- |
| **Time\_Bucket** | **Call\_Seconds (s)** |
| 10\_11 | 210.3246 |
| 11\_12 | 204.1766 |
| 12\_13 | 192.9231 |
| 13\_14 | 195.2707 |
| 14\_15 | 195.3203 |
| 15\_16 | 198.8873 |
| 16\_17 | 199.3751 |
| 17\_18 | 201.3928 |
| 18\_19 | 204.0942 |
| 19\_20 | 205.4205 |
| 20\_21 | 203.026 |
| 9\_10 | 199.4521 |

Graph Showing the rate of increase and decrease in call duration for each time bucket. The Orange Line is for the autowrapped calls and Blue line denotes the call duration count of the call wrapped by the Agents.



**Call Volume Analysis**

*Visualize the total number of calls received. This should be represented as a graph or chart showing the number of calls against time. Time should be represented in buckets (e.g., 1-2, 2-3, etc.).*

*Your Task: Can you create a chart or graph that shows the number of calls received in each time bucket?*

To analyses the Call Volume, I have performed the Group by based on the Time Bucket column. Moreover, I have further wanted to the type of call or Call Status too. So I plotted the graph for it too.

Code Used –

call\_count\_per\_bucket= df.groupby('Time\_Bucket').size().reset\_index(name='Call\_Count')

call\_count\_per\_bucket\_sort = call\_count\_per\_bucket.sort\_values(by= 'Time\_Bucket')

fig = px.bar(call\_count\_per\_bucket\_sort, x='Time\_Bucket', y='Call\_Count',

             title='Number of Calls Received in Each Time Bucket',

             labels={'Time\_Bucket': 'Time Bucket', 'Call\_Count': 'Number of Calls'})

# Show the plot

fig.show()

call\_count\_per\_bucket\_status = df.groupby(['Time\_Bucket', 'Call\_Status']).size().reset\_index(name='Call\_Count')

# Create grouped bar chart using Plotly Express

fig = px.bar(call\_count\_per\_bucket\_status, x='Time\_Bucket', y='Call\_Count', color='Call\_Status',

             title='Number of Calls Received in Each Time Bucket by Call Status',

             labels={'Time\_Bucket': 'Time Bucket', 'Call\_Count': 'Number of Calls', 'Call\_Status': 'Call Status'},

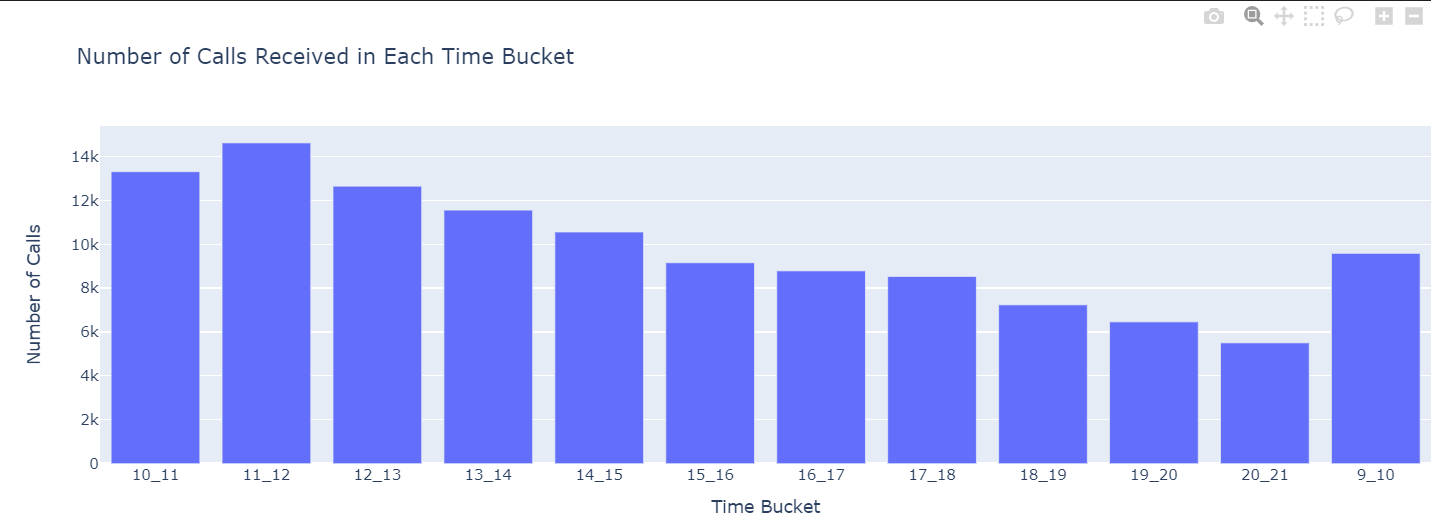
             barmode='group')

# Show the plot

fig.show()

call\_count\_per\_bucket\_status.to\_excel('call\_count\_per\_bucket\_status.xlsx')

The Number of call received in each bucket is as follows.



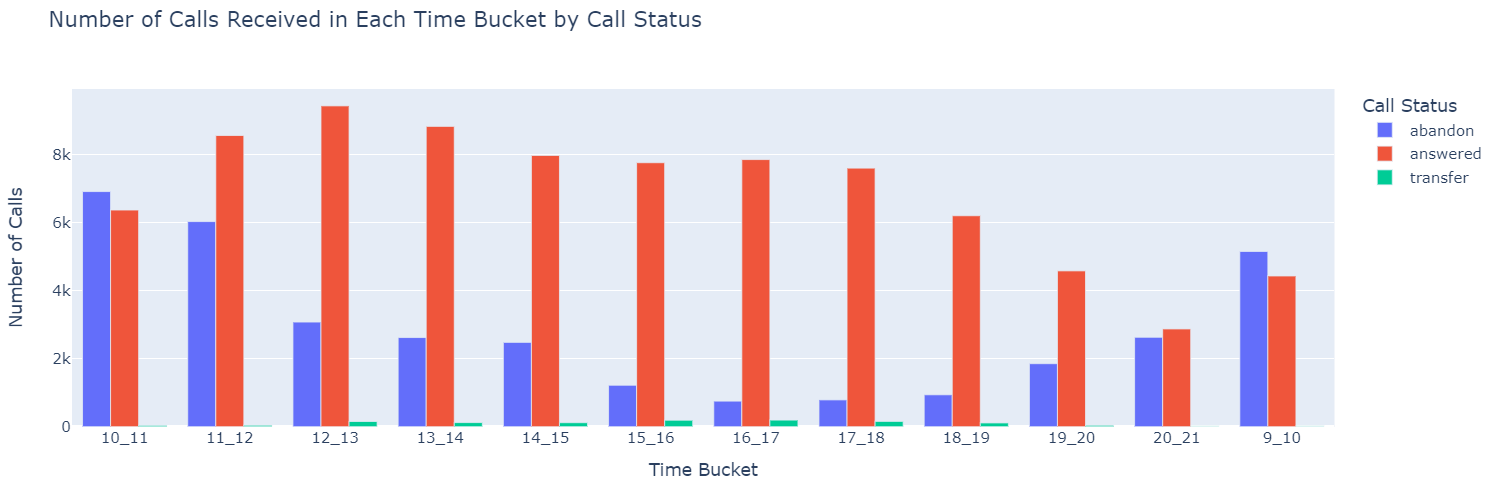
Further BreakDown of call recivied by Call Status. IN call status we recive three types of call –

Answered – Where we reply to customer

Abandon – No body took care of the customer and it can hamper our customer experience

Autowrapped – The Agent Automatically received the customer.

It will allow us to check how many calls that we received are taken care of or left unrevived. Further we can go on to improve this bin so that not all call goes unnoticed or not received.



|  |  |  |
| --- | --- | --- |
| **Time\_Bucket** | **Call\_Status** | **Call\_Count** |
| 10\_11 | abandon | 6911 |
| 10\_11 | answered | 6368 |
| 10\_11 | transfer | 34 |
| 11\_12 | abandon | 6028 |
| 11\_12 | answered | 8560 |
| 11\_12 | transfer | 38 |
| 12\_13 | abandon | 3073 |
| 12\_13 | answered | 9432 |
| 12\_13 | transfer | 147 |
| 13\_14 | abandon | 2617 |
| 13\_14 | answered | 8829 |
| 13\_14 | transfer | 115 |
| 14\_15 | abandon | 2475 |
| 14\_15 | answered | 7974 |
| 14\_15 | transfer | 112 |
| 15\_16 | abandon | 1214 |
| 15\_16 | answered | 7760 |
| 15\_16 | transfer | 185 |
| 16\_17 | abandon | 747 |
| 16\_17 | answered | 7852 |
| 16\_17 | transfer | 189 |
| 17\_18 | abandon | 783 |
| 17\_18 | answered | 7601 |
| 17\_18 | transfer | 150 |
| 18\_19 | abandon | 933 |
| 18\_19 | answered | 6200 |
| 18\_19 | transfer | 105 |
| 19\_20 | abandon | 1848 |
| 19\_20 | answered | 4578 |
| 19\_20 | transfer | 37 |
| 20\_21 | abandon | 2625 |
| 20\_21 | answered | 2870 |
| 20\_21 | transfer | 10 |
| 9\_10 | abandon | 5149 |
| 9\_10 | answered | 4428 |
| 9\_10 | transfer | 11 |

**Manpower Planning**

*The current rate of abandoned calls is approximately 30%. Propose a plan for manpower allocation during each time bucket (from 9 am to 9 pm) to reduce the abandon rate to 10%. In other words, you need to calculate the minimum number of agents required in each time bucket to ensure that at least 90 out of 100 calls are answered.*

*Your Task: What is the minimum number of agents required in each time bucket to reduce the abandon rate to 10%?*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time\_Bucket** | **Total\_agent\_per\_bucket** | **Total\_calls\_per\_bucket** | **Total\_abandon\_calls** | **abandon\_call\_percentage\_per\_bucket** | **agent\_require\_for\_10\_percent** |
| 10\_11 | 51 | 13313 | 6911 | 52 | 265 |
| 11\_12 | 59 | 14626 | 6028 | 41 | 242 |
| 12\_13 | 60 | 12652 | 3073 | 24 | 144 |
| 13\_14 | 58 | 11561 | 2617 | 23 | 133 |
| 14\_15 | 60 | 10561 | 2475 | 23 | 138 |
| 15\_16 | 58 | 9159 | 1214 | 13 | 75 |
| 16\_17 | 58 | 8788 | 747 | 9 | 52 |
| 17\_18 | 58 | 8534 | 783 | 9 | 52 |
| 18\_19 | 59 | 7238 | 933 | 13 | 77 |
| 19\_20 | 52 | 6463 | 1848 | 29 | 151 |
| 20\_21 | 27 | 5505 | 2625 | 48 | 130 |
| 9\_10 | 42 | 9588 | 5149 | 54 | 227 |
|  |  |  |  |  |  |

**How did I perform this Task**

**Step 1 –**

First I created the data frame that contain only abandon call status

Then I group by abandon calls data frame by the time bucket and count all the abandon calls that happen in each time frame.

abandon\_calls\_df = df[df['Call\_Status'] == 'abandon']

# Group by 'Time\_Bucket' and count the total number of abandon calls in each bucket

abandon\_call\_per\_bucket = abandon\_calls\_df.groupby('Time\_Bucket').size().reset\_index(name='Total\_abandon\_calls')

**Step 2 -**

Then I use original data frame that has all the type call status record group it with time bucket and and group it by the time bucket. But I also take all the unique agents count present in each data frame to understand how many agents are available to handle calls in each time bucket. Also I take count of call status to know the number of that we receive for each time bucket.

# Group by 'Time\_Bucket' and count the total number of calls and agents in each bucket

call\_count\_per\_bucket = df.groupby('Time\_Bucket').agg({'Agent\_Name': 'nunique', 'Call\_Status': 'count'}).reset\_index()

call\_count\_per\_bucket = call\_count\_per\_bucket.rename(columns={'Agent\_Name': 'Total\_agent\_per\_bucket', 'Call\_Status': 'Total\_calls\_per\_bucket'})

**Step 3 –**

It includes join the two data frame the abandon call per bucket data frame and call count per bucket data frame due to which I get following columns-

Time Bucket – that has time bucket

Total agent per bucket – count of unique agent present to take call in each bucket

Total calls per bucket – Total call received in this time bracket

Total abandon calls – Total call that we miss or could not attend too.

|  |  |  |  |
| --- | --- | --- | --- |
| **Time\_Bucket** | **Total\_agent\_per\_bucket** | **Total\_calls\_per\_bucket** | **Total\_abandon\_calls** |
| 10\_11 | 51 | 13313 | 6911 |
| 11\_12 | 59 | 14626 | 6028 |
| 12\_13 | 60 | 12652 | 3073 |
| 13\_14 | 58 | 11561 | 2617 |
| 14\_15 | 60 | 10561 | 2475 |
| 15\_16 | 58 | 9159 | 1214 |
| 16\_17 | 58 | 8788 | 747 |
| 17\_18 | 58 | 8534 | 783 |
| 18\_19 | 59 | 7238 | 933 |
| 19\_20 | 52 | 6463 | 1848 |
| 20\_21 | 27 | 5505 | 2625 |
| 9\_10 | 42 | 9588 | 5149 |

**Step 4 –**

merged\_df['abandon\_call\_percentage\_per\_bucket'] = round(100 \* (merged\_df['Total\_abandon\_calls'] / merged\_df['Total\_calls\_per\_bucket']), 0)

In this step we are calculating the abandon call for each time bucket and created new column anbandon call percentage per bucket .

Total abandon calls / Total calls per bucket \* 100

We get percentage of abandon call per bucket and there mean comes out to be 29 – 30 percent. Out task is to make this too 10 % and found out how many employee we will need to do so.

|  |
| --- |
| **abandon\_call\_percentage\_per\_bucket** |
| 52 |
| 41 |
| 24 |
| 23 |
| 23 |
| 13 |
| 9 |
| 9 |
| 13 |
| 29 |
| 48 |
| 54 |

**Step 4 –**

merged\_df['agent\_require\_for\_10\_percent'] = round(merged\_df['Total\_agent\_per\_bucket'] \* (merged\_df['abandon\_call\_percentage\_per\_bucket'] /10),0)

print(f'\nMean percentage of the abandon calls: {round(merged\_df["abandon\_call\_percentage\_per\_bucket"].mean(), 0)}')

merged\_df.to\_excel('agent\_require\_to\_10\_abandon.xlsx')

merged\_df['agent\_require\_for\_10\_percent'] = round(merged\_df['Total\_agent\_per\_bucket'] \* (merged\_df['abandon\_call\_percentage\_per\_bucket'] /10),0)

print(f'\nMean percentage of the abandon calls: {round(merged\_df["abandon\_call\_percentage\_per\_bucket"].mean(), 0)}')

merged\_df.to\_excel('agent\_require\_to\_10\_abandon.xlsx')

In this step we are calculating how many agents we need to bring abandon rate to 10 for each time bucket. Here is the logic which I used –

No Of agent required for the 10 percent abandon rate =

(Current rate (‘Abandon Call percentage per bucket)/ Desired rate (10)) \* Current number of agents

Also, I round the result to zero decimal points. The above table contain whole result that you can check.

**Code –**

# Filter out abandon calls

abandon\_calls\_df = df[df['Call\_Status'] == 'abandon']

# Group by 'Time\_Bucket' and count the total number of abandon calls in each bucket

abandon\_call\_per\_bucket = abandon\_calls\_df.groupby('Time\_Bucket').size().reset\_index(name='Total\_abandon\_calls')

# Group by 'Time\_Bucket' and count the total number of calls and agents in each bucket

call\_count\_per\_bucket = df.groupby('Time\_Bucket').agg({'Agent\_Name': 'nunique', 'Call\_Status': 'count'}).reset\_index()

call\_count\_per\_bucket = call\_count\_per\_bucket.rename(columns={'Agent\_Name': 'Total\_agent\_per\_bucket', 'Call\_Status': 'Total\_calls\_per\_bucket'})

# Merge the two DataFrames

merged\_df = pd.merge(call\_count\_per\_bucket, abandon\_call\_per\_bucket, on='Time\_Bucket', how='left')

# Calculate the abandon call percentage per bucket

merged\_df['abandon\_call\_percentage\_per\_bucket'] = round(100 \* (merged\_df['Total\_abandon\_calls'] / merged\_df['Total\_calls\_per\_bucket']), 0)

# Calculate the number of agents required to achieve a 10% abandon rate in each bucket

merged\_df['agent\_require\_for\_10\_percent'] = round(merged\_df['Total\_agent\_per\_bucket'] \* (merged\_df['abandon\_call\_percentage\_per\_bucket'] /10),0)

print(f'\nMean percentage of the abandon calls: {round(merged\_df["abandon\_call\_percentage\_per\_bucket"].mean(), 0)}')

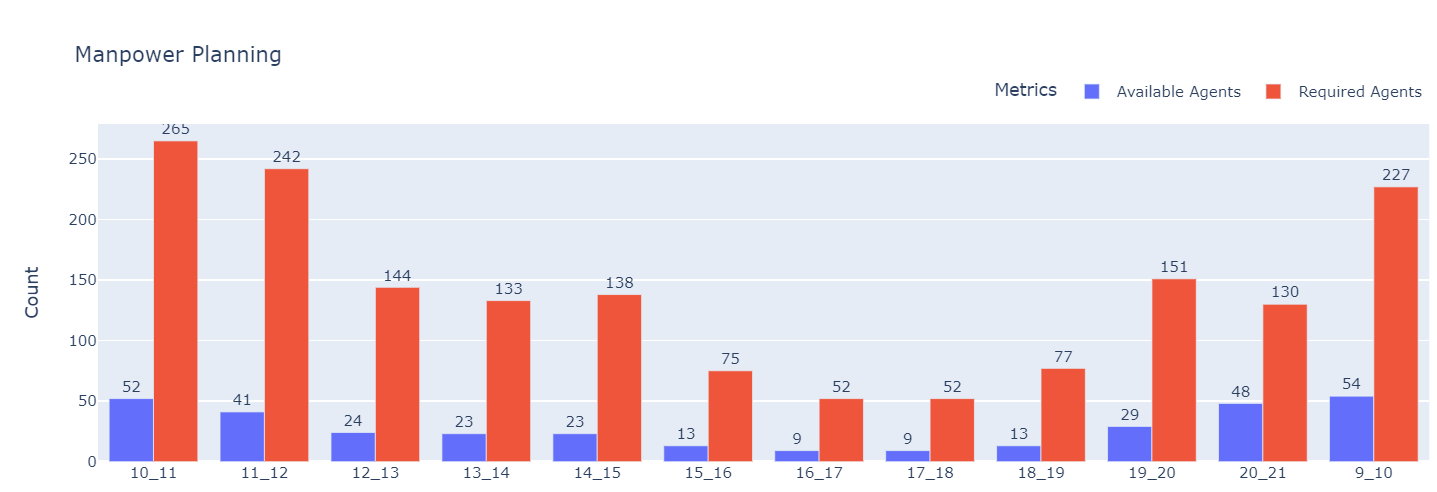
merged\_df.to\_excel('agent\_require\_to\_10\_abandon.xlsx')

merged\_df

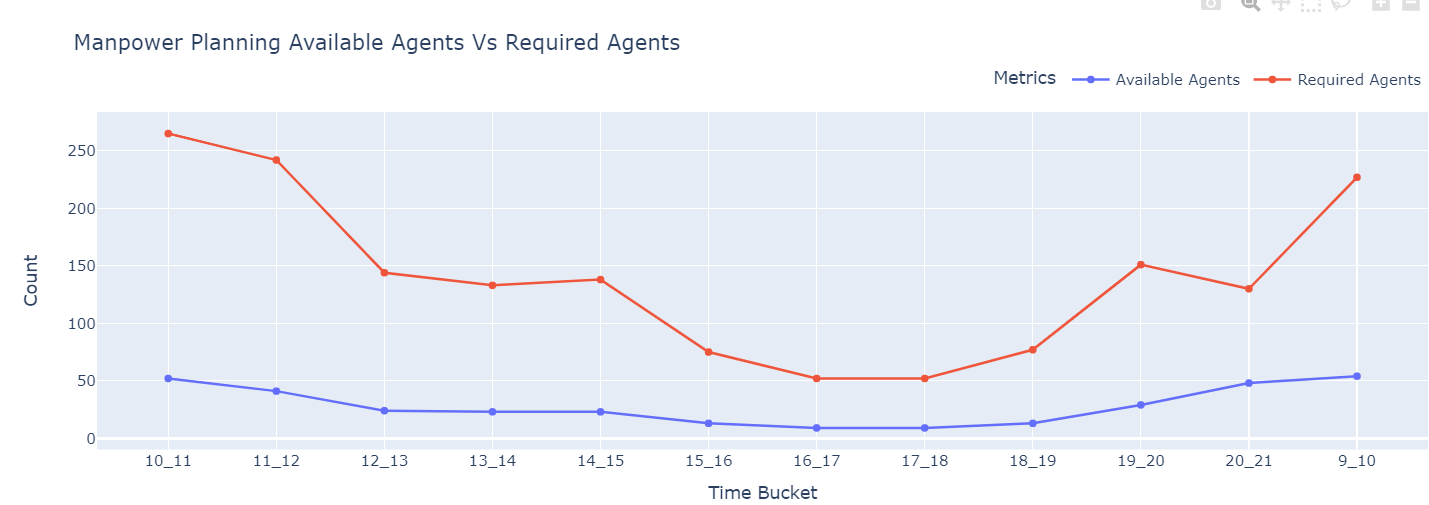
**Here is plot visualizations –**

**Manpower Planning Available agent and Agent**

Manpower Planning Available agent and Agent we require to bring the abandon call rate to 10 % .

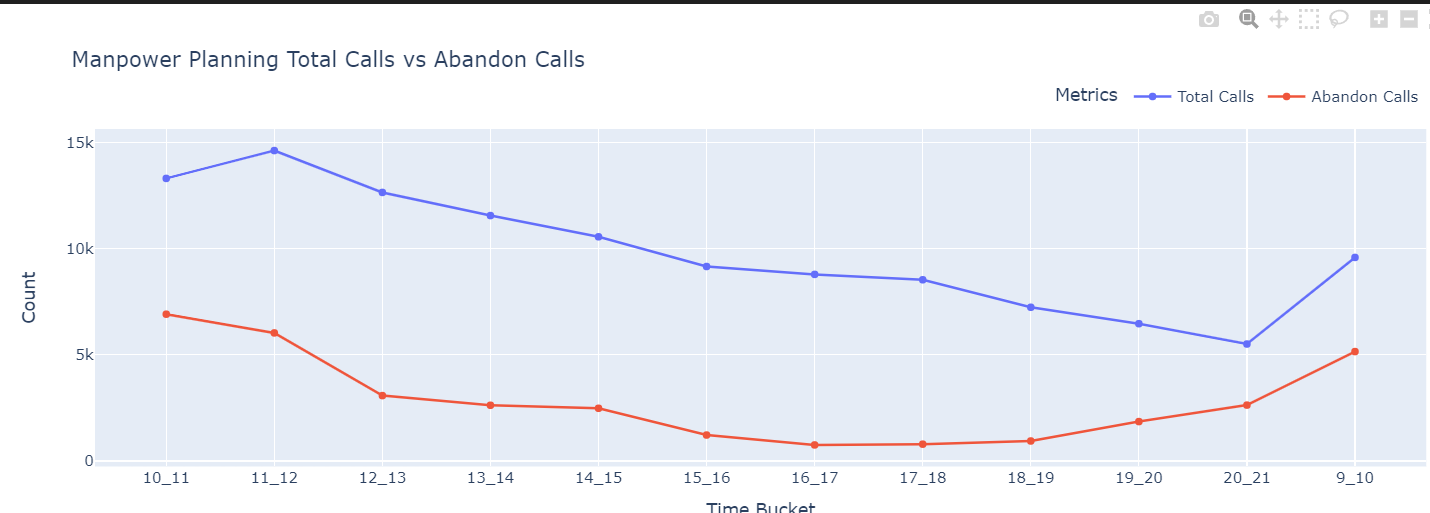


**ManPower Line chart Available Agents vs Required Agents**



**Manpower Planning**

**Total Calls vs Abandon Calls**



**Night Shift Manpower Planning**

*Customers also call ABC Insurance Company at night but don't get an answer because there are no agents available. This creates a poor customer experience. Assume that for every 100 calls that customers make between 9 am and 9 pm, they also make 30 calls at night between 9 pm and 9 am. The distribution of these 30 calls is as follows:*

*Your Task: Propose a manpower plan for each time bucket throughout the day, keeping the maximum abandon rate at 10%.*

**How did I perform this task –**

First I group the data by day .

monthly\_grouped = df.groupby(pd.Grouper(key='Date\_&\_Time', freq='D'))

**Step 1 –**

monthly\_summary = monthly\_grouped.agg({

    'Call\_Status': 'count',

    'Agent\_Name': 'nunique',

    'Call\_Seconds (s)' : 'mean'

}).reset\_index()

Then I created new data frame monthly summary that includes total count of the calls and count of all the unique agent’s name and also the average of call duration.

**Step 2 –**

monthly\_summary = monthly\_summary.rename(columns={'Call\_Status': 'total\_calls\_a\_day', 'Agent\_Name': 'Number\_of\_employees\_working'})

monthly\_summary['total\_calls\_at\_night'] =round(monthly\_summary['total\_calls\_a\_day']\*0.3)

Then I added the new column which holds the 30 % of count of total call and the column is called total call at night. Since it is mention for total of 100 calls the 30 calls comes in the night so the rate of night call are 30% of total call.

**Step 3 –**

monthly\_summary['Calls\_to\_answer\_for\_10%\_abandon\_rate\_at\_night'] = round(monthly\_summary['total\_calls\_at\_night']\*0.90)

monthly\_summary['total\_hours\_needed'] = monthly\_summary['Average\_duration'] \* monthly\_summary['Calls\_to\_answer\_for\_10%\_abandon\_rate\_at\_night'] / 3600

I want to answer the 90% of the calls that comes in the night. For that I took the 90 % of the calls at night (total call at night) column. And created new column called calls to answer for 10% abandon rate at night.

Since I have average of call duration per call in the data frame, multiplying it with the calls to answer for 10% abandon rate at night and dividing it with 3600 will give me hours needed to attend to these calls. And I created a new column called ‘Total hours needed’.

**Step 4 –**

According to the problem a candidate can has 9-hour shift. Out of which 1.5 they use for lunch and refreshment, which gives only 7.5 hours of working time. Apart from which the 60% of 7.5 an employee use to attend call, which 4.5 hours of working time. Dividing it by the total hours needed we can get number of employees we need to get the 90 % answer rate or 10 % abandon rate at night.

available\_hours\_per\_agent = 0.6 \* (9 - 1.5)  # 60% of (total working hours - lunch time)

# Calculate the number of agents needed to handle the calls

monthly\_summary['number\_of\_agents\_needed'] = round(monthly\_summary['total\_hours\_needed'] / available\_hours\_per\_agent)

# Print the updated DataFrame

monthly\_summary.to\_excel('night\_manpower.xlsx')

**Here is how whole table looks –**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date\_&\_Time** | **total\_calls\_a\_day** | **Number\_of\_employees\_working** | **Average\_duration** | **total\_calls\_at\_night** | **Calls\_to\_answer\_for\_10%\_abandon\_rate\_at\_night** | **total\_hours\_needed** | **number\_of\_agents\_needed** |
| 2022-01-01 00:00:00 | 4644 | 35 | 145.707149 | 1393 | 1254 | 50.7546569 | 11 |
| 2022-01-02 00:00:00 | 3351 | 35 | 171.2930469 | 1005 | 904 | 43.01358732 | 10 |
| 2022-01-03 00:00:00 | 4789 | 44 | 169.7354354 | 1437 | 1293 | 60.96331054 | 14 |
| 2022-01-04 00:00:00 | 5113 | 48 | 168.5793076 | 1534 | 1381 | 64.66889552 | 14 |
| 2022-01-05 00:00:00 | 4790 | 50 | 176.7845511 | 1437 | 1293 | 63.49511795 | 14 |
| 2022-01-06 00:00:00 | 4951 | 45 | 167.4490002 | 1485 | 1336 | 62.14218452 | 14 |
| 2022-01-07 00:00:00 | 4948 | 40 | 152.9949475 | 1484 | 1336 | 56.77812494 | 13 |
| 2022-01-08 00:00:00 | 4672 | 39 | 157.4152397 | 1402 | 1262 | 55.18278682 | 12 |
| 2022-01-09 00:00:00 | 3652 | 30 | 148.1782585 | 1096 | 986 | 40.58437857 | 9 |
| 2022-01-10 00:00:00 | 4983 | 42 | 156.2791491 | 1495 | 1346 | 58.43103742 | 13 |
| 2022-01-11 00:00:00 | 4637 | 41 | 169.4451154 | 1391 | 1252 | 58.92924568 | 13 |
| 2022-01-12 00:00:00 | 4643 | 38 | 152.9041568 | 1393 | 1254 | 53.26161462 | 12 |
| 2022-01-13 00:00:00 | 4123 | 34 | 167.6740238 | 1237 | 1113 | 51.83921902 | 12 |
| 2022-01-14 00:00:00 | 3155 | 36 | 178.8358162 | 946 | 851 | 42.27479988 | 9 |
| 2022-01-15 00:00:00 | 3058 | 38 | 181.9054938 | 917 | 825 | 41.68667566 | 9 |
| 2022-01-16 00:00:00 | 5142 | 40 | 131.1540257 | 1543 | 1389 | 50.6035949 | 11 |
| 2022-01-17 00:00:00 | 22347 | 49 | 42.31507585 | 6704 | 6034 | 70.9247688 | 16 |
| 2022-01-18 00:00:00 | 5774 | 41 | 137.9923796 | 1732 | 1559 | 59.75836662 | 13 |
| 2022-01-19 00:00:00 | 4703 | 44 | 159.5300872 | 1411 | 1270 | 56.27866964 | 13 |
| 2022-01-20 00:00:00 | 4322 | 39 | 175.7549745 | 1297 | 1167 | 56.97390425 | 13 |
| 2022-01-21 00:00:00 | 3675 | 38 | 174.1102041 | 1102 | 992 | 47.97703401 | 11 |
| 2022-01-22 00:00:00 | 3291 | 42 | 188.8717715 | 987 | 888 | 46.5883703 | 10 |
| 2022-01-23 00:00:00 | 3225 | 39 | 171.7516279 | 968 | 871 | 41.5543522 | 9 |

**Complete Code –**

import pandas as pd

total\_working\_days\_per\_week = 6

unplanned\_leaves\_per\_month = 4

total\_working\_hours = 9

break\_time = 1.5

call\_handling\_percentage = 0.6  # 60%

total\_days\_in\_month = 30

# Calculate total calls during the day and at night

total\_calls\_day = 100

total\_calls\_night = 30

# Group by day

monthly\_grouped = df.groupby(pd.Grouper(key='Date\_&\_Time', freq='D'))

# Aggregate total calls, total agents, and other metrics per month

monthly\_summary = monthly\_grouped.agg({

    'Call\_Status': 'count',

    'Agent\_Name': 'nunique',

    'Call\_Seconds (s)' : 'mean'

}).reset\_index()

# Rename columns

monthly\_summary = monthly\_summary.rename(columns={'Call\_Status': 'total\_calls\_a\_day', 'Agent\_Name': 'Number\_of\_employees\_working','Call\_Seconds (s)' : 'Average\_duration'})

monthly\_summary['total\_calls\_at\_night'] =round( monthly\_summary['total\_calls\_a\_day']\*0.3)

monthly\_summary['Calls\_to\_answer\_for\_10%\_abandon\_rate\_at\_night'] = round(monthly\_summary['total\_calls\_at\_night']\*0.90)

monthly\_summary['total\_hours\_needed'] = monthly\_summary['Average\_duration'] \* monthly\_summary['Calls\_to\_answer\_for\_10%\_abandon\_rate\_at\_night'] / 3600

available\_hours\_per\_agent = 0.6 \* (9 - 1.5)  # 60% of (total working hours - lunch time)

# Calculate the number of agents needed to handle the calls

monthly\_summary['number\_of\_agents\_needed'] = round(monthly\_summary['total\_hours\_needed'] / available\_hours\_per\_agent)

# Print the updated DataFrame

monthly\_summary.to\_excel('night\_manpower.xlsx')

monthly\_summary