Name: Nikhil Zodape

Email Id.: Nikszodape@gmail.com

Project Name: ABC Call Volume Trend Analysis

Using: Python and Excel

Problem Statement:

The attached dataset is of Inbound Calls of an ABC company from the insurance category consist of a Customer Experience (CX) Inbound calling team for 23 days. Data includes Agent_Name, Agent_ID, Queue_Time [duration for which customer have to wait before they get connected to anagent], Time [time at which call was made by customer in a day], Time_Bucket [for easiness we have also provided you with the time bucket], Duration [duration for which a customer and executives are on call], Call_Seconds [for simplicity we have also converted those time into seconds], callstatus (Abandon, answered, transferred).

A customer experience (CS) tram consists of professionals who analyze customer feedback and data, and share insights with the rest of the organization. Typically, these teams fulfil various roles and responsibilities such as: Customer experience programs (CX programs), Digital Customer Experience, Design and process, Internal Communications, Voice of the customer (Voc), user experiences, Customer experience management, Journey mapping, Nurturing customer interactions, Customer success, Customer support, Handling customer data, Learning about the customer journey.

Interactive Voice Response (IVR), Robotic process Automation (RPA), Predictive Analysis, Intelligent Routing and some other customer experience tools we can use in this project.

One of the key roles in a CX team is that of the customer service representative, also known as a call center agent. These agents handle various types of support, including email, inbound, outbound, and social media support.

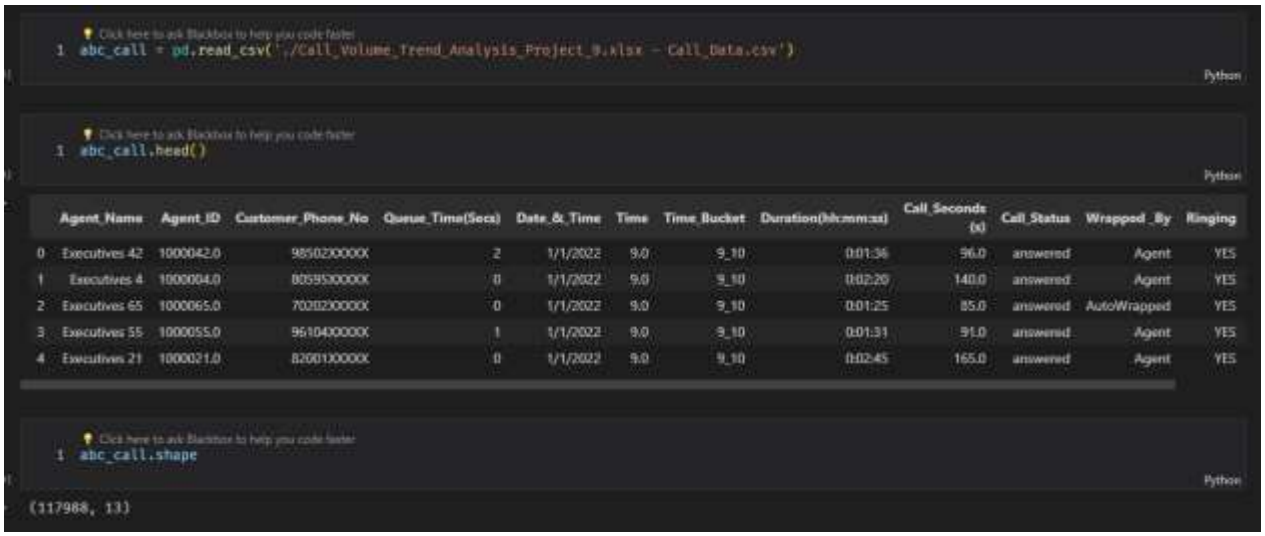
Inbound customer support, which is the focus of this project, involves handling incoming calls from existing or prospective customers. The goal is to attract, engage, and delight customers, turning them into loyal advocates for the business.

Business Understanding:

Advertising is a marketing strategy used to increase sales and raise awareness of a business's products or services. It helps form first impressions of a customer before they make a purchase. Businesses can target various audiences, including local, regional, national, or international, using various methods such as internet directories, trade press, radio, cinema, outdoor advertising, national papers, magazines, and TV. The advertising industry is highly competitive, with many players bidding heavily for the same audience. Companies must use their analytical skills to effectively target these audiences across various media platforms, converting them into customers at a low cost.

Dataset Information:

- The Dataset named **Call_Data.csv** has **117988 rows** and **13 columns**.



```
abc_call = pd.read_csv('./Call_Volume_Trend_Analysis_Project_9.xlsx - Call_Data.csv')
```

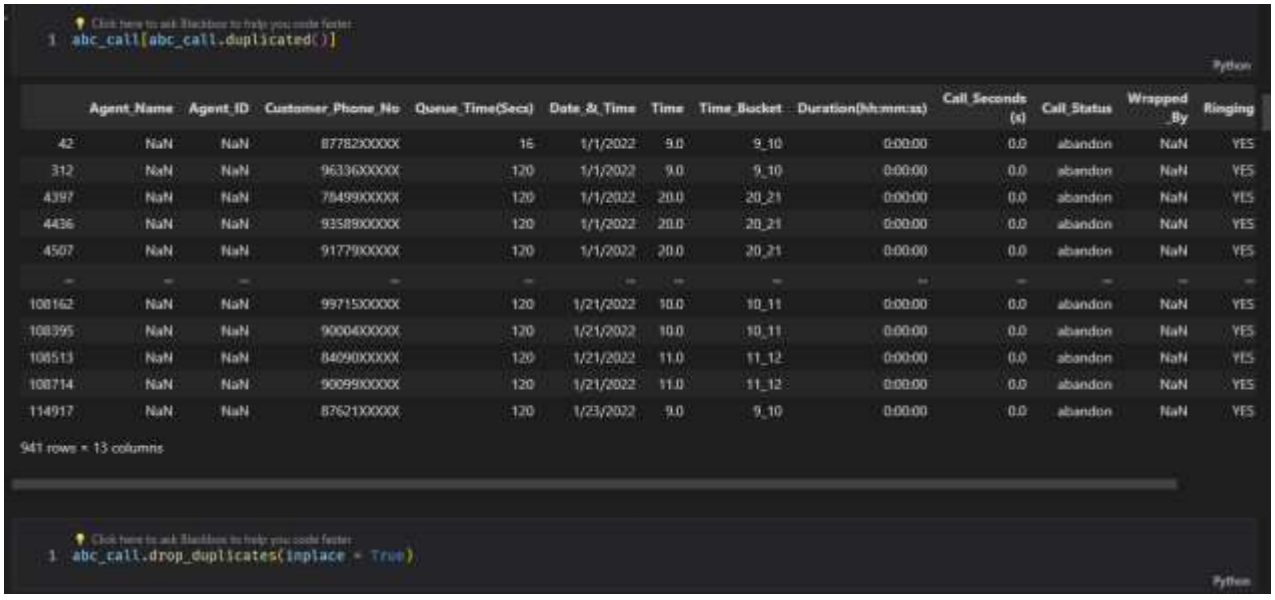
```
abc_call.head()
```

	Agent_Name	Agent_ID	Customer_Phone_No	Queue_Time(Secs)	Date_&_Time	Time	Time_Bucket	Duration(hh:mm:ss)	Call_Seconds (s)	Call_Status	Wrapped_By	Ringing
0	Executives 42	1000042.0	9850200000X	2	1/1/2022	9.0	9_10	0:01:36	96.0	answered	Agent	YES
1	Executives 4	1000004.0	8059530000X	0	1/1/2022	9.0	9_10	0:02:20	140.0	answered	Agent	YES
2	Executives 65	1000065.0	7020290000X	0	1/1/2022	9.0	9_10	0:01:25	85.0	answered	AutoWrapped	YES
3	Executives 55	1000055.0	9610400000X	1	1/1/2022	9.0	9_10	0:01:31	91.0	answered	Agent	YES
4	Executives 21	1000021.0	8200100000X	0	1/1/2022	9.0	9_10	0:02:45	165.0	answered	Agent	YES

```
abc_call.shape
```

```
(117988, 13)
```

- Data Pre-Processing
  - **Checking for Duplicates values** and Dropping those duplicates values from the dataset.



```
abc_call[abc_call.duplicated()]
```

	Agent_Name	Agent_ID	Customer_Phone_No	Queue_Time(Secs)	Date_&_Time	Time	Time_Bucket	Duration(hh:mm:ss)	Call_Seconds (s)	Call_Status	Wrapped_By	Ringing
42	NaN	NaN	8778200000X	16	1/1/2022	9.0	9_10	0:00:00	0.0	abandon	NaN	YES
312	NaN	NaN	9633600000X	120	1/1/2022	9.0	9_10	0:00:00	0.0	abandon	NaN	YES
4397	NaN	NaN	7849900000X	120	1/1/2022	20.0	20_21	0:00:00	0.0	abandon	NaN	YES
4436	NaN	NaN	9358900000X	120	1/1/2022	20.0	20_21	0:00:00	0.0	abandon	NaN	YES
4507	NaN	NaN	9177900000X	120	1/1/2022	20.0	20_21	0:00:00	0.0	abandon	NaN	YES
...	...	...	...	...	...	...	...	...	...	...	...	...
108162	NaN	NaN	9971500000X	120	1/21/2022	10.0	10_11	0:00:00	0.0	abandon	NaN	YES
108395	NaN	NaN	9000400000X	120	1/21/2022	10.0	10_11	0:00:00	0.0	abandon	NaN	YES
108513	NaN	NaN	8409000000X	120	1/21/2022	11.0	11_12	0:00:00	0.0	abandon	NaN	YES
108714	NaN	NaN	9009900000X	120	1/21/2022	11.0	11_12	0:00:00	0.0	abandon	NaN	YES
114917	NaN	NaN	8762100000X	120	1/23/2022	9.0	9_10	0:00:00	0.0	abandon	NaN	YES

941 rows × 13 columns

```
abc_call.drop_duplicates(inplace = True)
```

- Checking total numbers of rows we got after dropping the duplicates values from the dataset.

```
abc_call.shape
```

```
(117047, 13)
```

- **Checking for Null Values:** Created the null_col dataframe and sort the values in descending order to get only Null values.

```
null_col = pd.DataFrame()
null_col['Null'] = abc_call.isnull().sum().sort_values(ascending = False)
null_col = null_col[null_col['Null'] > 0]
null_col
```

	Null
Wrapped _By	46936
Agent_Name	33257
Agent_ID	33257

- We found that all the rows where **Agent_Name** and **Agent_ID** was **Null** are rows denoting **abandoned** calls.

```
abc_call[abc_call['Agent_Name'].isna()]['Agent_ID'].unique()
```

```
array([nan])
```

```
abc_call[abc_call['Agent_ID'].isna()]['Agent_Name'].unique()
```

```
array([nan], dtype=object)
```

```
abc_call[abc_call['Agent_ID'].isna()]['Wrapped _By'].unique()
```

```
array([nan], dtype=object)
```

```
abc_call[abc_call['Agent_ID'].isna()]['Call_Status'].unique()
```

```
array(['abandon'], dtype=object)
```

```
abc_call[abc_call['Wrapped _By'].isna()]['Call_Status'].unique()
```

```
array(['abandon', 'answered', 'transfer'], dtype=object)
```

- Some calls where **Wrapped _By** was **Null** were **answered** or **transferred** calls. So replaced them with value '**Agent**'.
- Rest of the **Null** values were replaced by value '**Not Available**'

```
abc_call['Wrapped _By'].value_counts()
```

```
Agent          60396
AutoWrapped     9715
Name: Wrapped _By, dtype: int64
```

```
abc_call[(abc_call['Wrapped _By'].isna()) & ((abc_call['Call_Status'] == 'answered') |
                                             (abc_call['Call_Status'] == 'transfer'))].fillna("Agent", inplace = True)
```

```
abc_call.fillna('Not Available', inplace=True)
```

```
abc_call.isna().sum().sort_values(ascending = False)
```

```
Agent_Name           0
Agent_ID             0
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          0
Ringing              0
IVR _Duration        0
dtype: int64
```

- Now we will check for any errors in the columns

```
obj_col = list(abc_call.dtypes[abc_call.dtypes == 'object'].index)
print(obj_col)

for col in obj_col:
    print(col, '\n', abc_call[col].unique())

# Assuming abc_call is a DataFrame, you may want to replace 'df' with 'abc_call' in the remaining code
for col in [obj_col[2], obj_col[3], obj_col[4], obj_col[6], obj_col[7], obj_col[8]]:
    print(col, '\n', abc_call[col].unique())

print(obj_col[5], '(Hour)', '\n', np.sort(pd.Series(abc_call[obj_col[5]].unique()).str.split(':').apply(lambda x: x[0]).unique()))
print(obj_col[5], '(Minute)', '\n', np.sort(pd.Series(abc_call[obj_col[5]].unique()).str.split(':').apply(lambda x: x[1]).unique()))
print(obj_col[5], '(Second)', '\n', np.sort(pd.Series(abc_call[obj_col[5]].unique()).str.split(':').apply(lambda x: x[2]).unique()))

print(obj_col[9], '(Hour)','\n', np.sort(pd.Series(abc_call[obj_col[9]].unique()).str.split(':').apply(lambda x : x[0]).unique()))
print(obj_col[9], '(Minute)', '\n', np.sort(pd.Series(abc_call[obj_col[9]].unique()).str.split(':').apply(lambda x: x[1]).unique()))
print(obj_col[9], '(Second)', '\n', np.sort(pd.Series(abc_call[obj_col[9]].unique()).str.split(':').apply(lambda x: x[2]).unique()))
```

['Agent_Name', 'Agent_ID', 'Customer_Phone_No', 'Date_&_Time', 'Time_Bucket', 'Duration(hh:mm:ss)', 'Call_Status', 'Wrapped _By', 'Ringing', 'IVR _Duration']

Agent_Name
['Executives 42' 'Executives 4' 'Executives 65' 'Executives 55' 'Executives 21' 'Not Available' 'Executives 49' 'Executives 50' 'Executives 59' 'Executives 16' 'Executives 60' 'Executives 6' 'Executives 51' 'Executives 40' 'Executives 54' 'Executives 41' 'Executives 15' 'Executives 10' 'Executives 31' 'Executives 46' 'Executives 22' 'Executives 26' 'Executives 53' 'Executives 23' 'Executives 47' 'Executives 37' 'Executives 58' 'Executives 24' 'Executives 38' 'Executives 18' 'Executives 48' 'Executives 9' 'Executives 30' 'Executives 7' 'Executives 39' 'Executives 13'

'Executives 62' 'Executives 19' 'Executives 35' 'Executives 25' 'Executives 29' 'Executives 14' 'Executives 34' 'Executives 33' 'Executives 43' 'Executives 1' 'Executives 27' 'Executives 12' 'Executives 28' 'Executives 52' 'Executives 36' 'Executives 5' 'Executives 3' 'Executives 17' 'Executives 63' 'Executives 8' 'Executives 61' 'Executives 44' 'Executives 45' 'Executives 64' 'Executives 57' 'Executives 2' 'Executives 11' 'Executives 20' 'Executives 56' 'Executives 32']
Agent_ID
[1000042.0 1000004.0 1000065.0 1000055.0 1000021.0 'Not Available' 1000049.0 1000050.0 1000059.0 1000016.0 1000060.0 1000006.0 1000051.0 1000040.0 1000054.0 1000041.0 1000015.0 1000010.0 1000031.0 1000046.0 1000022.0 1000026.0 1000053.0 1000023.0 1000047.0 1000037.0 1000058.0 1000024.0 1000038.0 1000018.0 1000048.0 1000009.0 1000030.0 1000007.0 1000039.0 1000013.0 1000062.0 1000019.0 1000035.0 1000025.0 1000029.0 1000014.0 1000034.0 1000033.0 1000043.0 1000001.0 1000027.0 1000012.0 1000028.0 1000052.0 1000036.0 1000005.0 1000003.0 1000017.0 1000063.0 1000008.0 1000061.0 1000044.0 1000045.0 1000064.0 1000057.0 1000002.0 1000011.0 1000020.0 1000056.0 1000032.0]
Customer_Phone_No
['98502XXXXX' '80595XXXXX' '70202XXXXX' ... '95511XXXXX' '62921XXXXX' '77184XXXXX']
Date_&_Time
['1/1/2022' '1/2/2022' '1/3/2022' '1/4/2022' '1/5/2022' '1/6/2022' '1/7/2022' '1/8/2022' '1/9/2022' '1/10/2022' '1/11/2022' '1/12/2022' '1/13/2022' '1/14/2022' '1/15/2022' '1/16/2022' '1/17/2022' '1/18/2022' '1/19/2022' '1/20/2022' '1/21/2022' '1/22/2022' '1/23/2022']
Time_Bucket
['9_10' '10_11' '11_12' '12_13' '13_14' '14_15' '15_16' '16_17' '17_18' '18_19' '19_20' '20_21']
Duration(hh:mm:ss)
['0:01:36' '0:02:20' '0:01:25' ... '0:15:55' '0:19:45' '0:16:08']
Call_Status
['answered' 'abandon' 'transfer']
Wrapped _By
['Agent' 'AutoWrapped' 'Not Available']
Ringing
['YES']
IVR _Duration
['0:00:16' '0:00:26' '0:00:25' '0:00:23' '0:00:13' '0:00:17' '0:00:20' '0:00:44' '0:00:15' '0:00:40' '0:00:42' '0:00:19' '0:00:18' '0:00:48' '0:00:45' '0:00:21' '0:00:41' '0:00:46' '0:00:28' '0:00:39' '0:01:44' '0:00:49' '0:00:14' '0:00:24' '0:01:02' '0:00:54' '0:00:36' '0:00:22' '0:00:52' '0:00:12' '0:00:27' '0:01:09' '0:00:43' '0:00:11' '0:00:50' '0:00:00' '0:01:07' '0:01:06' '0:00:33' '0:00:51' '0:01:34' '0:01:10' '0:00:34' '0:01:17' '0:00:37' '0:00:47' '0:01:38' '0:01:12' '0:01:11' '0:00:57' '0:01:04' '0:00:53' '0:00:35' '0:00:38' '0:01:14' '0:01:37' '0:01:01' '0:01:16' '0:01:15' '0:01:35' '0:01:08' '0:01:33' '0:01:32' '0:00:59' '0:01:13' '0:01:58' '0:01:27' '0:02:08' '0:00:55' '0:01:24' '0:01:03' '0:01:40' '0:01:19' '0:02:55' '0:00:10' '0:01:05' '0:01:20' '0:00:08' '0:00:32' '0:00:56' '0:02:09' '0:03:18' '0:00:29' '0:01:00' '0:05:58' '0:01:59' '0:01:36' '0:01:43' '0:02:25' '0:01:31' '0:01:30' '0:01:41' '0:01:29' '0:03:16' '0:04:34' '0:01:18' '0:00:30' '0:02:48' '0:04:12' '0:02:06' '0:02:51' '0:01:51' '0:00:31' '0:01:45' '0:01:57' '0:01:26' '0:05:28' '0:01:23' '0:00:58' '0:02:53' '0:01:56' '0:02:03' '0:02:24' '0:01:42' '0:01:39' '0:02:21' '0:02:26' '0:02:01' '0:01:21' '0:02:07' '0:03:44' '0:03:42' '0:02:52' '0:02:59' '0:02:02' '0:01:25' '0:02:04' '0:02:00' '0:02:22' '0:00:09' '0:01:28' '0:01:22' '0:02:34' '0:02:23' '0:56:48' '0:01:55' '0:08:47' '0:02:30' '0:02:35' '0:01:47' '0:02:54' '0:05:12' '0:03:43' '0:03:45' '0:02:28' '0:02:32' '0:02:27' '0:02:29' '0:17:23' '0:02:57' '0:03:01' '0:05:25' '0:02:20' '0:01:48' '0:03:11' '0:02:45' '0:02:36' '0:06:15' '0:07:37' '0:04:58' '0:02:05' '0:01:53' '0:01:46' '0:10:10' '0:02:49' '0:10:09' '0:03:25' '0:03:39' '0:03:23' '0:09:27' '0:02:10' '0:03:17' '0:04:38' '0:04:03']
Customer_Phone_No
['98502XXXXX' '80595XXXXX' '70202XXXXX' ... '95511XXXXX' '62921XXXXX' '77184XXXXX']
Date_&_Time

['1/1/2022' '1/2/2022' '1/3/2022' '1/4/2022' '1/5/2022' '1/6/2022' '1/7/2022' '1/8/2022' '1/9/2022' '1/10/2022' '1/11/2022' '1/12/2022' '1/13/2022' '1/14/2022' '1/15/2022' '1/16/2022' '1/17/2022' '1/18/2022' '1/19/2022' '1/20/2022' '1/21/2022' '1/22/2022' '1/23/2022']
Call_Status
['answered' 'abandon' 'transfer']
Wrapped _By
['Agent' 'AutoWrapped' 'Not Available']
Ringing
['YES']
Duration(hh:mm:ss) (Hour)
['0' '1']
Duration(hh:mm:ss) (Minute)
['00' '01' '02' '03' '04' '05' '06' '07' '08' '09' '10' '11' '12' '13' '14' '15' '16' '17' '18' '19' '20' '21' '22' '23' '24' '25' '26' '27' '28' '29' '30' '31' '32' '33' '34' '35' '36' '37' '38' '39' '40' '41' '43' '44' '47' '48' '52' '53']
Duration(hh:mm:ss) (Second)
['00' '01' '02' '03' '04' '05' '06' '07' '08' '09' '10' '11' '12' '13' '14' '15' '16' '17' '18' '19' '20' '21' '22' '23' '24' '25' '26' '27' '28' '29' '30' '31' '32' '33' '34' '35' '36' '37' '38' '39' '40' '41' '42' '43' '44' '45' '46' '47' '48' '49' '50' '51' '52' '53' '54' '55' '56' '57' '58' '59']
IVR _Duration (Hour)
['0']
IVR _Duration (Minute)
['00' '01' '02' '03' '04' '05' '06' '07' '08' '09' '10' '17' '56']
IVR _Duration (Second)
['00' '01' '02' '03' '04' '05' '06' '07' '08' '09' '10' '11' '12' '13' '14' '15' '16' '17' '18' '19' '20' '21' '22' '23' '24' '25' '26' '27' '28' '29' '30' '31' '32' '33' '34' '35' '36' '37' '38' '39' '40' '41' '42' '43' '44' '45' '46' '47' '48' '49' '50' '51' '52' '53' '54' '55' '56' '57' '58' '59']

- **Dealing with the error in the customer_phone_no.:**

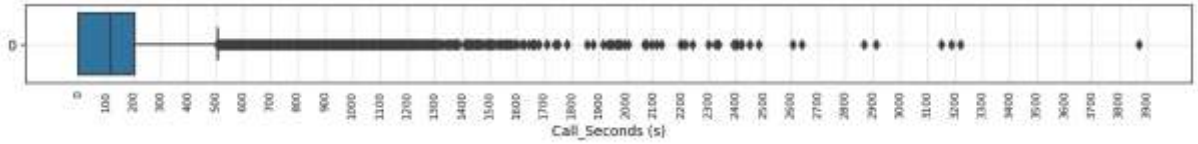
- As the below process showed that **Customer_Phone_No** column had row(s) with **5 digit number** which is an **error**.
- So we checked and found **one** such row and replaced the column value with '**XXXXXXXXXX**'

- **Dealing with Outliers:**
  - **Queue_Time(s):** From below we can see that this distribution is not having any outlies in it.

- **Time:** From below we can see there are no outlies in time columns also.

Time [9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21.]

- **Call_Seconds (s):** From below observation we can see there are outliers in this columns.



To deal with this outliers we have selected the threshold of 3000 and on further investigation we didn't find anything unusual. So we didn't do any changes in the data.

```
abc_call[abc_call['Call_Seconds (s)'] > 3000]
```

	Agent_Name	Agent_ID	Customer_Phone_No	Queue_Time(Secs)	Date_& _Time	Time	Time_Bucket	Duration(hh:mm:ss)	Call_Seconds (s)	Call_Status	Wrapped_By	Ringing	IVR_Duration
49457	Executives 10	1000010.0	7908400000(	4	1/11/2022	17.0	17_18	1:04:32	3872.0	answered	Agent	YES	0:00:24
49844	Executives 43	1000043.0	9896100000(	13	1/11/2022	18.0	18_19	0:52:29	3149.0	answered	Agent	YES	0:00:14
52597	Executives 5	1000005.0	9880000000(	24	1/12/2022	13.0	13_14	0:53:05	3185.0	answered	Agent	YES	0:00:13
91887	Executives 39	1000039.0	9764800000(	108	1/17/2022	18.0	18_19	0:53:39	3219.0	answered	AutoWrapped	YES	0:00:24

- Dealing with the Inconsistencies in some columns:

```
dur_sec = pd.Series(abc_call['Duration(hh:mm:ss)'].str.split(':').apply(lambda x: sum([a*b for a,b in zip([3600,60,1], map(int, x))])))
call_sec = abc_call['Call_Seconds (s)']
```

```
abc_call[dur_sec - call_sec) != 0.0 ]
```

Agent_Name	Agent_ID	Customer_Phone_No	Queue_Time(Secs)	Date_& _Time	Time	Time_Bucket	Duration(hh:mm:ss)	Call_Seconds (s)	Call_Status	Wrapped_By	Ringing	IVR_Duration

```
abc_call['Call_Status'].unique()
```

```
array(['answered', 'abandon', 'transfer'], dtype=object)
```

```
abc_call[abc_call['Call_Status'] == 'abandon']['Call_Seconds (s)'].unique()
```

```
array([0.])
```

```
abc_call[abc_call['Call_Status'] == 'abandon']['Wrapped _By'].unique()
```

```
array(['Not Available'], dtype=object)
```

```
Click here to ask Blackbox to help you code faster
group = abc_call.groupby('Agent_ID')['Call_Status'].value_counts().unstack().reset_index()
group.fillna(0, inplace = True)
group['received'] = group['answered'] + group['transfer']
group.drop(['answered', 'transfer'], axis = 1, inplace = True)
group = group[group['received'] == 0.0]
group
```

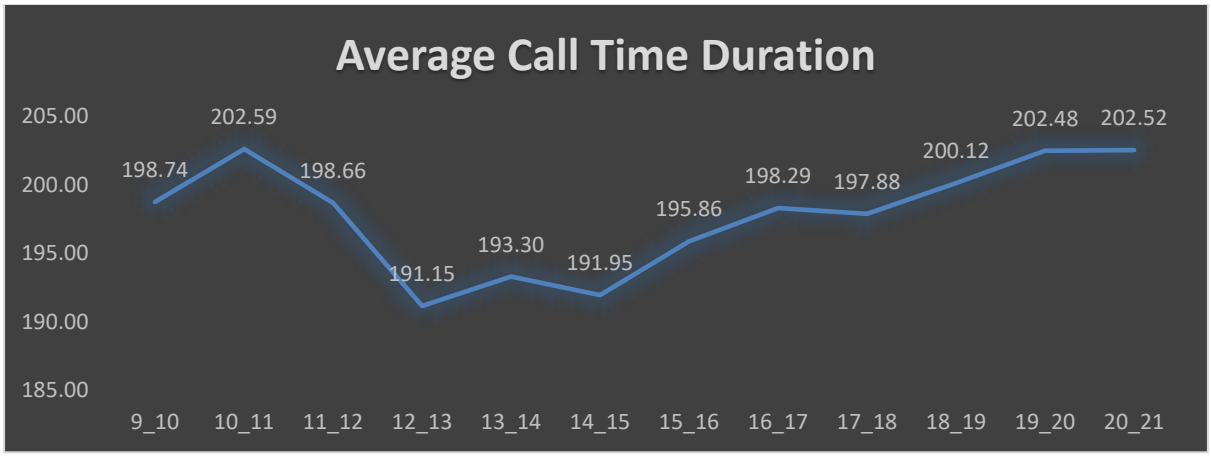
Call_Status	Agent_ID	abandon	received
1	1000002.0	1.0	0.0
10	1000011.0	1.0	0.0
13	1000014.0	1.0	0.0
43	1000044.0	1.0	0.0
44	1000045.0	1.0	0.0
56	1000057.0	1.0	0.0
63	1000064.0	1.0	0.0
65	Not Available	33257.0	0.0

```
Click here to ask Blackbox to help you code faster
ag_id = list(group['Agent_ID'].values[:-1])
ag_nm = [abc_call[abc_call['Agent_ID'] == i] ['Agent_Name'].values[0] for i in ag_id]
abc_call['Agent_ID'].replace(ag_id, 'Not Available', inplace = True)
abc_call['Agent_Name'].replace(ag_nm, "Not Available", inplace = True)
```

Analysis:

1. **Average Call Duration:** Determine the average distribution of all incoming calls received by agents. This should be calculated for each time buckets.

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

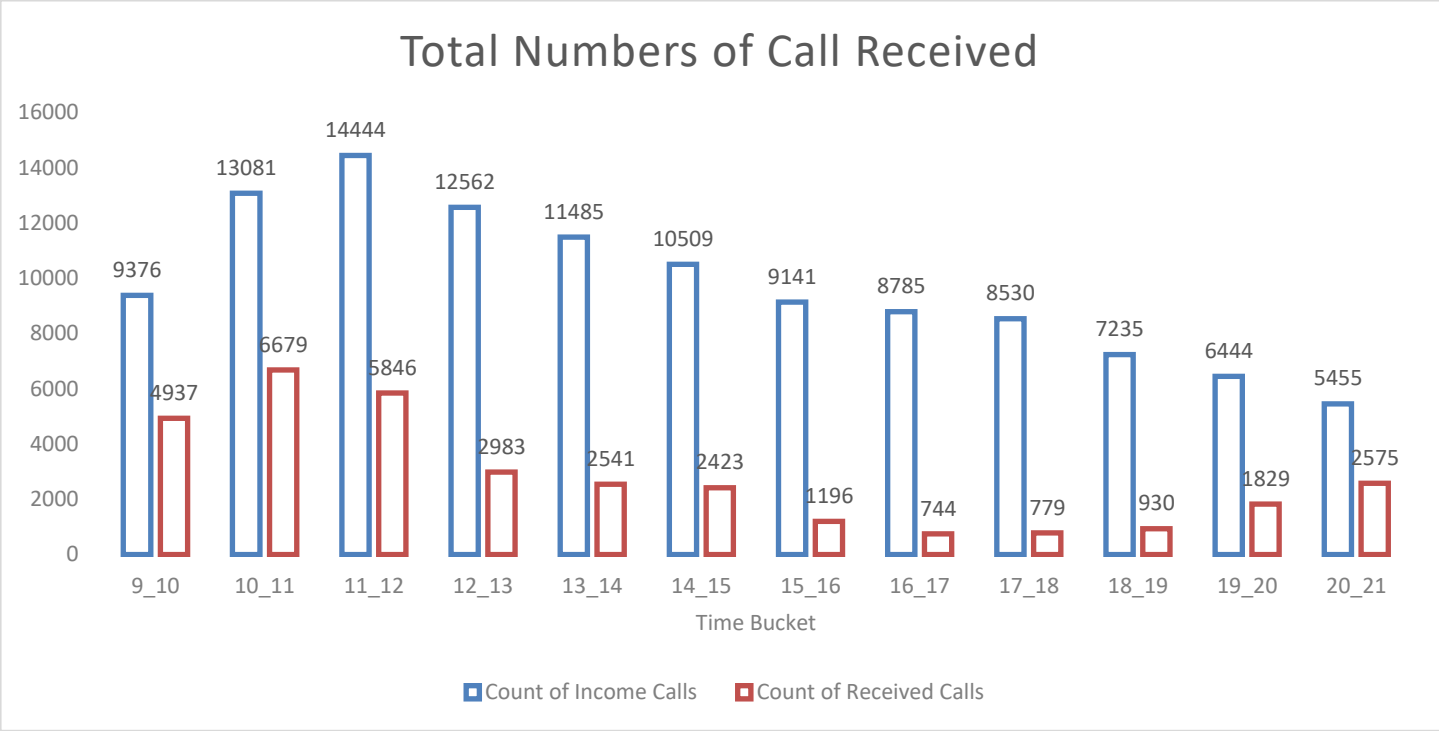


Average Call Time Duration: The highest average call duration was observed in the time buckets 10 to 11, 19 to 20, and 21 to 21, with an average duration of 197.80 seconds.

On the other hand, the time bucket 12 to 13 had the lowest average call duration.

2. **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?



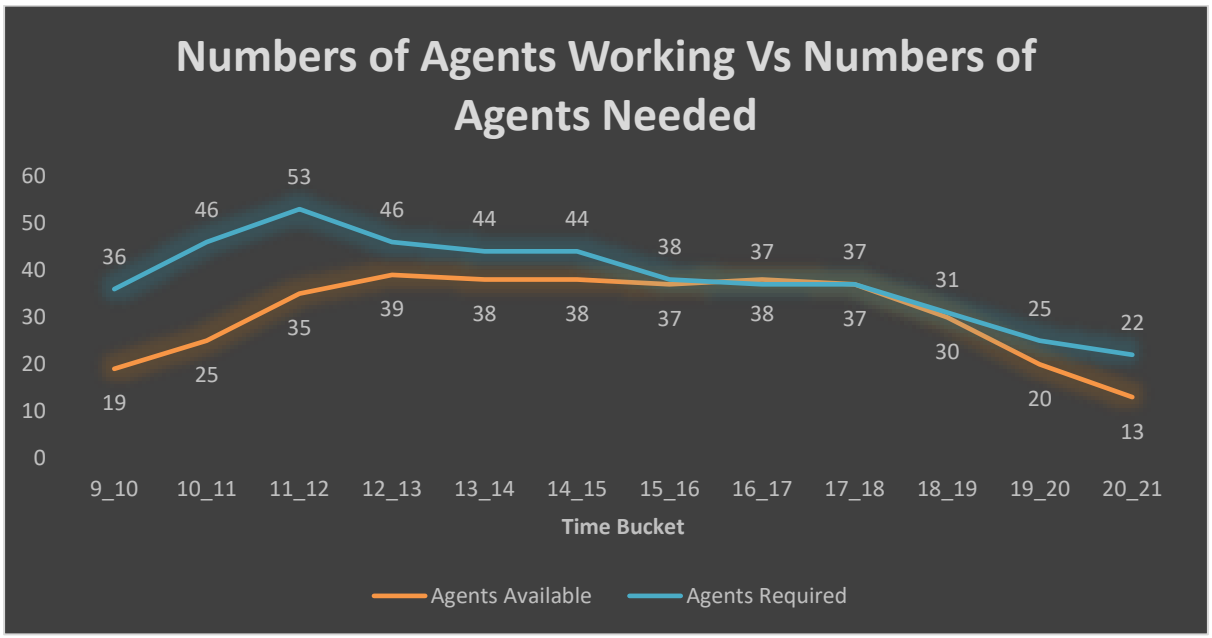
- We can observe that the number of **received** calls received **first increases** with time **before dropping down**.
- We can also observe that the number of **abandoned** calls is **very high** in the **morning hours** and as the day progresses, the number of **abandoned calls reduces**.

Assumptions & Calculations

Work Hours :	9
Break :	1.5
Actual Working Hours :	7.5
Occupancy :	60%
Total Working Seconds as per Occupancy:	16200
Average Call Time/Agent :	197.80
Call Capacity of an Agent/day :	81
Call Capacity of an Agent/Hour :	18

3. **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%?

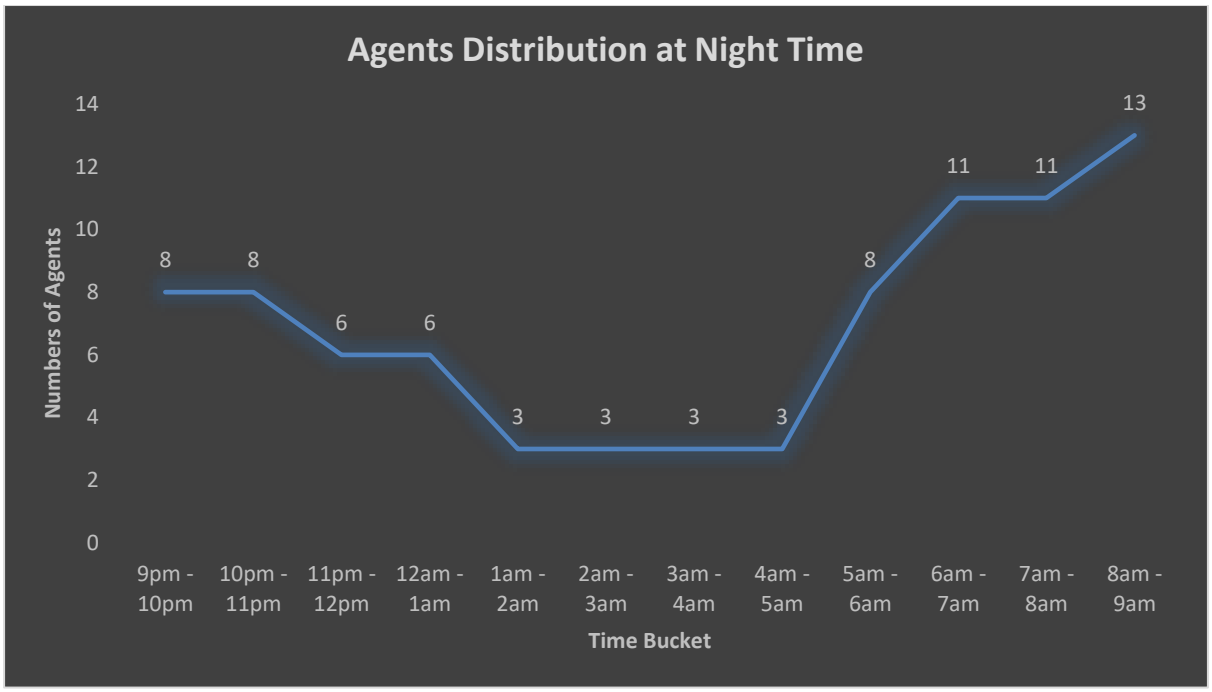


- We can observe that to maintain a maximum of **10% abandon rate**, we need to **increase** the availability of agents in the **morning hours** by a **large margin** as in these hours, the number of incoming calls is quite high and the number of agents available currently are quite low.
- During **afternoon hours** and during **late evening hours**, we need to **increase** the availability of agents by a **slight margin** to maintain a maximum of **10% abandon rate**.

4. **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%.

Distribution of 30 calls coming in night for every 100 calls coming in between 9am - 9pm (i.e. 12 hrs slot)											
9pm- 10pm	10pm - 11pm	11pm- 12am	12am- 1am	1am - 2am	2am - 3am	3am - 4am	4am - 5am	5am - 6am	6am - 7am	7am - 8am	8am - 9am
3	3	2	2	1	1	1	1	3	4	4	5



Night-time Call Volume and Agents required: It was observed that for every 100 calls made by customers during the 9:00 a.m. to 9:00 p.m. timeframe, there were also 30 calls made during the night from 9:00 p.m. to 9:00 a.m. To handle these night-time calls, a manpower of 22 unique agents would be needed.

Link of complete Python workbook:
https://drive.google.com/file/d/1V3XKsWuq6JY3gcYUMp3oXGkL3tcvfvkj/view?usp=drive_link

Link of complete Excel workbook:
https://docs.google.com/spreadsheets/d/1CRMq97HSSJVSWf7D6QcY5_iHYCggrlbi/edit?usp=drive_link&ouid=107933073293773244015&rtpof=true&sd=true

Video Link:
https://drive.google.com/file/d/15JM3br-KvH66SPOPm9diAVe3gRpaiboi/view?usp=drive_link