In [89]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [25]:
```
# Importing Data
call_center=pd.read_excel("Call Center.xlsx")
call_center
```

Out[25]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Call Id	Agent	Date	Time	Topic	Answered (Y/N)	Resolved	Speed of answer in seconds	AvgTalkDuration	Satisfaction rating
2	ID0001	Diane	2021-01-01	09:12:58	Contract related	Y	Y	109	00:02:23	3
3	ID0002	Becky	2021-01-01	09:12:58	Technical Support	Y	N	70	00:04:02	3
4	ID0003	Stewart	2021-01-01	09:47:31	Contract related	Y	Y	10	00:02:11	3
...	...	...	...	...	...	...	...	...	...	...
4997	ID4996	Jim	2021-03-31	16:37:55	Payment related	Y	Y	22	00:05:40	1
4998	ID4997	Diane	2021-03-31	16:45:07	Payment related	Y	Y	100	00:03:16	3
4999	ID4998	Diane	2021-03-31	16:53:46	Payment related	Y	Y	84	00:01:49	4
5000	ID4999	Jim	2021-03-31	17:02:24	Streaming	Y	Y	98	00:00:58	5
5001	ID5000	Diane	2021-03-31	17:39:50	Contract related	N	N	NaN	NaN	NaN

5002 rows × 10 columns

In [26]:
```
#duplicating data for cleaning
cl=call_center.copy()
```

In [27]:
```
cl
```

Out[27]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Call Id	Agent	Date	Time	Topic	Answered (Y/N)	Resolved	Speed of answer in seconds	AvgTalkDuration	Satisfaction rating
2	ID0001	Diane	2021-01-01	09:12:58	Contract related	Y	Y	109	00:02:23	3
3	ID0002	Becky	2021-01-01	09:12:58	Technical Support	Y	N	70	00:04:02	3
4	ID0003	Stewart	2021-01-01	09:47:31	Contract related	Y	Y	10	00:02:11	3
...	...	...	...	...	...	...	...	...	...	...
4997	ID4996	Jim	2021-03-31	16:37:55	Payment related	Y	Y	22	00:05:40	1
4998	ID4997	Diane	2021-03-31	16:45:07	Payment related	Y	Y	100	00:03:16	3
4999	ID4998	Diane	2021-03-31	16:53:46	Payment related	Y	Y	84	00:01:49	4
5000	ID4999	Jim	2021-03-31	17:02:24	Streaming	Y	Y	98	00:00:58	5
5001	ID5000	Diane	2021-03-31	17:39:50	Contract related	N	N	NaN	NaN	NaN

5002 rows × 10 columns

In [31]:
```
#making row 1 the table header
cl.columns = cl.iloc[0]
cl = cl[1:].reset_index(drop=True)
```

In [32]:
```
cl
```

Out[32]:

	Call Id	Agent	Date	Time	Topic	Answered (Y/N)	Resolved	Speed of answer in seconds	AvgTalkDuration	Satisfaction rating
0	ID0001	Diane	2021-01-01	09:12:58	Contract related	Y	Y	109	00:02:23	3
1	ID0002	Becky	2021-01-01	09:12:58	Technical Support	Y	N	70	00:04:02	3
2	ID0003	Stewart	2021-01-01	09:47:31	Contract related	Y	Y	10	00:02:11	3
3	ID0004	Greg	2021-01-01	09:47:31	Contract related	Y	Y	53	00:00:37	2
4	ID0005	Becky	2021-01-01	10:00:29	Payment related	Y	Y	95	00:01:00	3
...	...	...	...	...	...	...	...	...	...	...
4995	ID4996	Jim	2021-03-31	16:37:55	Payment related	Y	Y	22	00:05:40	1
4996	ID4997	Diane	2021-03-31	16:45:07	Payment related	Y	Y	100	00:03:16	3
4997	ID4998	Diane	2021-03-31	16:53:46	Payment related	Y	Y	84	00:01:49	4
4998	ID4999	Jim	2021-03-31	17:02:24	Streaming	Y	Y	98	00:00:58	5
4999	ID5000	Diane	2021-03-31	17:39:50	Contract related	N	N	NaN	NaN	NaN

5000 rows × 10 columns

In [118]:
```
#displaying all rows
cl = pd.DataFrame(cl)
pd.set_option('display.max_rows', None)
```

In [121]:
```
#display 10 rows
pd.set_option('display.max_rows', 10)
```

In [122]:
```
#fill nan value with 0
cl.fillna(0, inplace=True)
cl
```

Out[122]:

	Call Id	Agent	Date	Time	Topic	Answered (Y/N)	Resolved	Speed of answer in seconds	AvgTalkDuration	Satisfaction rating	Month
0	ID0001	Diane	2021-01-01	09:12:58	Contract related	Y	Y	109	00:02:23	3	January
1	ID0002	Becky	2021-01-01	09:12:58	Technical Support	Y	N	70	00:04:02	3	January
2	ID0003	Stewart	2021-01-01	09:47:31	Contract related	Y	Y	10	00:02:11	3	January
3	ID0004	Greg	2021-01-01	09:47:31	Contract related	Y	Y	53	00:00:37	2	January
4	ID0005	Becky	2021-01-01	10:00:29	Payment related	Y	Y	95	00:01:00	3	January
...	...	...	...	...	...	...	...	...	...	...	...
4995	ID4996	Jim	2021-03-31	16:37:55	Payment related	Y	Y	22	00:05:40	1	March
4996	ID4997	Diane	2021-03-31	16:45:07	Payment related	Y	Y	100	00:03:16	3	March
4997	ID4998	Diane	2021-03-31	16:53:46	Payment related	Y	Y	84	00:01:49	4	March
4998	ID4999	Jim	2021-03-31	17:02:24	Streaming	Y	Y	98	00:00:58	5	March
4999	ID5000	Diane	2021-03-31	17:39:50	Contract related	N	N	0	0	0	March

5000 rows × 11 columns

In [40]:
```
#confirming all adjustment done on data
cl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Call Id                    5000 non-null   object
 1   Agent                      5000 non-null   object
 2   Date                       5000 non-null   object
 3   Time                       5000 non-null   object
 4   Topic                      5000 non-null   object
 5   Answered (Y/N)             5000 non-null   object
 6   Resolved                   5000 non-null   object
 7   Speed of answer in seconds  5000 non-null   int64
 8   AvgTalkDuration            5000 non-null   object
 9   Satisfaction rating        5000 non-null   int64
dtypes: int64(2), object(8)
memory usage: 390.8+ KB
```

In [41]:
```
cl["Agent"].unique()
```

```
Out[41]:array(['Diane', 'Becky', 'Stewart', 'Greg', 'Jim', 'Joe', 'Martha', 'Dan'],
      dtype=object)
```

In [42]:
```
cl["Topic"].unique()
```

```
Out[42]:array(['Contract related', 'Technical Support', 'Payment related',
       'Admin Support', 'Streaming'], dtype=object)
```

In [43]:
```
cl["Satisfaction rating"].unique()
```

```
Out[43]:array([3, 2, 0, 4, 5, 1])
```

Data Analysis:

In [123]:
```
#distribution of call volumes over time(date and time)
trend=cl.groupby(['Date','Time'])['Call Id'].count()
pd.DataFrame(trend)
```

Out[123]:

Date	Time	Call Id
2021-01-01	09:12:58	2
	09:47:31	2
	10:00:29	2
	10:22:05	2
	11:13:55	2
...	...	...
2021-03-31	16:37:55	1
	16:45:07	1
	16:53:46	1
	17:02:24	1
	17:39:50	1

2421 rows × 1 columns

In [111]:
```
#the proportion of answered and unresolved calls?
answered= (cl['Answered (Y/N)'] == 'Y').sum()
unresolved=(cl['Resolved']=='N').sum()
answered
#called answered =4054
unresolved
#unresolved is:1354
proportion=(unresolved/answered)*100
rounded=round(proportion,0)
print('The proportion of answered and unresolved call is :%',rounded)
```

```
The proportion of answered and unresolved call is :% 33.0
```

Having %33 of answered but unresolved calls is unsatisfactory; Agents need improvement so as to ensure customer satisfaction.

In [55]:
```
#distribution of call topics
pd.DataFrame(cl.groupby(['Topic'])['Topic'].count().sort_values(ascending=False))
```

Out[55]:

	Topic
Topic	
Streaming	1022
Technical Support	1019
Payment related	1007
Admin Support	976
Contract related	976

In [114]:
```
#distribution of customer satisfaction rating
cl.groupby(['Satisfaction rating','Agent'])['Agent'].count().unstack()
```

Out[114]:

Agent	Becky	Dan	Diane	Greg	Jim	Joe	Martha	Stewart
Satisfaction rating								
0	114	110	132	122	130	109	124	105
1	64	49	50	43	57	54	47	53
2	42	47	50	57	54	51	47	48
3	150	166	155	161	157	149	149	131
4	160	143	139	136	157	141	159	145
5	101	118	107	105	111	89	112	100

- Dan has more 5 and 3 start ratring records than other agents
- Becky takes the lead in 4 and 1 star rating
- Greg has the highest number of 2 start rating
- Diane has more 0 rating than other agents

How agents performance varies based on different metrics:

- Agents engagements
- Resolved and unresolved cases by agent
- number of answered and unanswered call by agent
- proportion of unresolved calls by agent
- proportion of unanswered call by agent

In [44]:
```
# Agents engagements
agents_entry=cl.groupby(['Agent'])['Agent'].count().sort_values(ascending=False)
pd.DataFrame(agents_entry)
```

Out[44]:

	Agent
Agent	
Jim	666
Martha	638
Dan	633
Diane	633
Becky	631
Greg	624
Joe	593
Stewart	582

In [48]:
```
#Resolved  and unresolved cases by agent
cl.groupby(['Resolved','Agent'])['Agent'].count().unstack()
```

Out[48]:

Agent	Becky	Dan	Diane	Greg	Jim	Joe	Martha	Stewart
Resolved								
N	169	162	181	169	181	157	177	158
Y	462	471	452	455	485	436	461	424

The table above shows Resolved and unresolved cases of all agents

- jim has the highest resolved cases
- jim and diane have the highest number of unresolved cases
- according to out[25],he picks more cases than other agents.

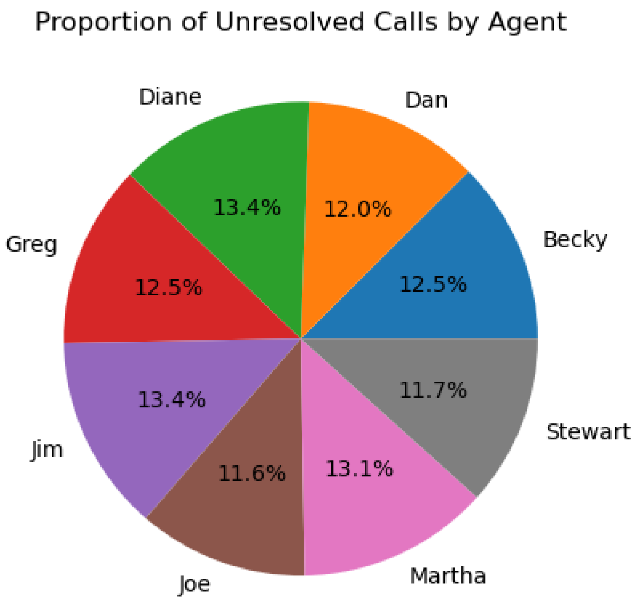
In [49]:
```
#number of answered and unanswered call
cl.groupby(['Answered (Y/N)','Agent'])['Agent'].count().unstack()
```

Out[49]:

Agent	Becky	Dan	Diane	Greg	Jim	Joe	Martha	Stewart
Answered (Y/N)								
N	114	110	132	122	130	109	124	105
Y	517	523	501	502	536	484	514	477

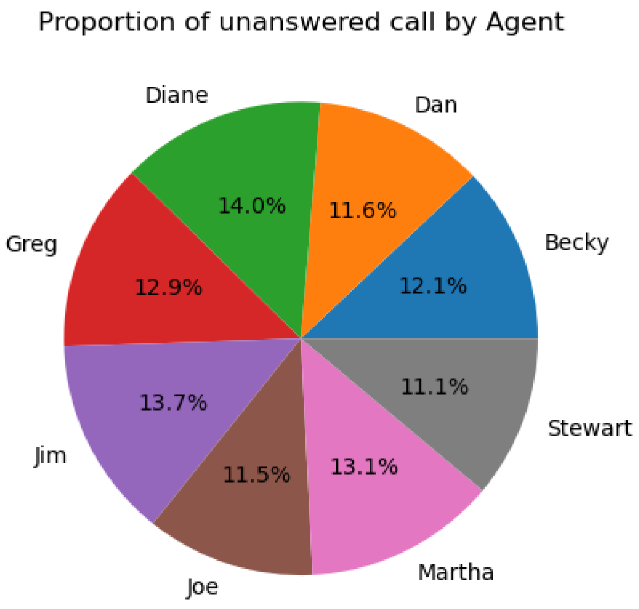
In [93]:
```
#proportion of unresolved calls by agent
unresolved_counts = cl[cl['Resolved'] == 'N'].groupby('Agent').size()

# Plot a pie chart
plt.pie(unresolved_counts, labels=unresolved_counts.index, autopct='%1.1f%%')
plt.title('Proportion of Unresolved Calls by Agent')
plt.show()
```



proportion of unresolved calls by agents are not customer friendly,all agents need to be more diligent at there duty

In [105]:
```
#proportion of unanswered call by agent
Unanswered_counts = cl[cl['Answered (Y/N)'] == 'N'].groupby('Agent').size()
# Plot a pie chart
plt.pie(Unanswered_counts, labels=Unanswered_counts.index, autopct='%1.1f%%')
plt.title('Proportion of unanswered call by Agent')
plt.show()
```



- Diane has the highest percentage of unswered calls
- proportion of unanswered calls by agent are not customer friendly,all agents need to be cautioned

Recommendation for improving call centre performance and customer satisfaction:

- performance evaluations and setting achievable goals for agents.working critically with customers rating,analizing these ratings so as to know the performance of each agent.
- Recognize and reward high-performing agents to boost morale and motivation within the team.
- Educate customers on self-service options and encourage them to use self-help resources,this would reduce the volume of calls handled by agents also would increase effectiveness.

In [ ]: