Dataset Description

Given a month's data of EdTech Company for analysis. This dataset contains the details of the leads in various stages of the customer acquisition flow.

Expected Outcome

1. Brainstorm and identify the right metrics and frame proper questions for analysis. Your analysis should help your

   a. Business team to understand the lead's journey and stages with scope for improvement

   b. Business heads to understand their team performance

   c. Managers to understand their target areas

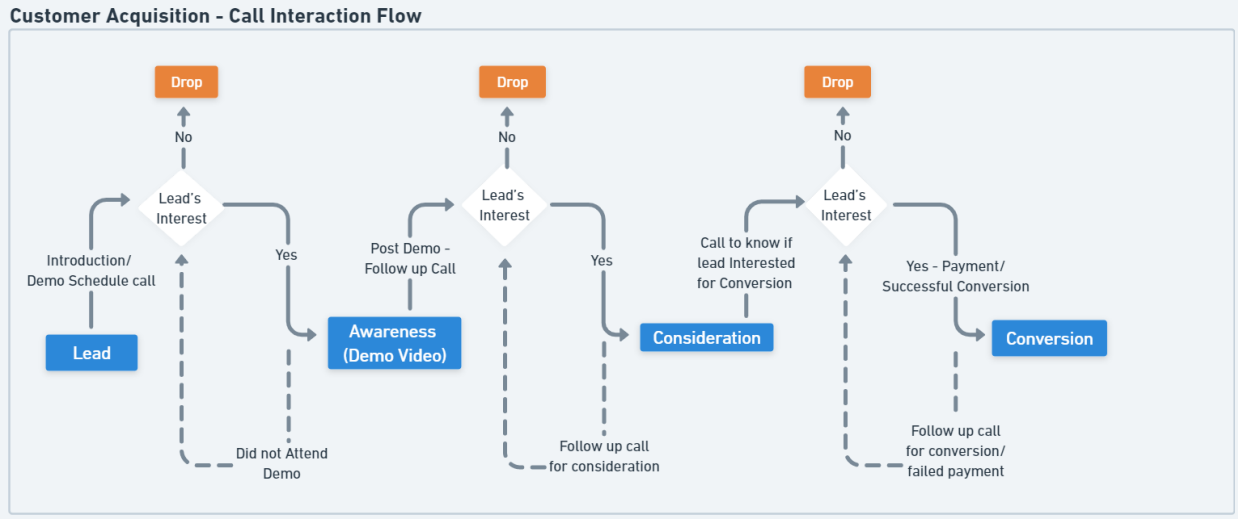
1. In case you identify any outliers in the data set, make a note of them and exclude them from your analysis.
2. Build the best suitable dashboard presenting your insights.

Tools to Use

• Visualization tools like Google Data Studio (preferred) or Tableau.

• BigQuery SQL if required, not mandatory.

Customer Acquisition Flow



Customer Acquisition Flow

Overview of the Dataset:

```
Customer Acquisition Key Stages:  Lead - Awareness - Consideration -
Conversion
```

Dataset provided contains:

```
* Basic details of leads

* Hierarchy structure of sales managers and their assigned leads

* Lead interaction details

* Demo watched details of leads

* Reason for not being interested
```

In [2]:
```
# Importing the required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
```

In [3]:
```
l_b_d = pd.read_csv("leads_basic_details.csv")
l_d_w_d = pd.read_csv("leads_demo_watched_details.csv")
l_i_d = pd.read_csv("leads_interaction_details.csv")
l_r_f_n_i = pd.read_csv("leads_reasons_for_no_interest.csv")
s_m_a_l_d = pd.read_csv("sales_managers_assigned_leads_details.csv");
```

Data Pre-processing

Dataset 1: leads_basic_details

In [4]:
```
l_b_d.shape
```

Out[4]:
```
(360, 7)
```

In [5]:
```
# Checking the info()
l_b_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 360 entries, 0 to 359
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   lead_id           360 non-null    object
 1   age               360 non-null    int64
 2   gender            360 non-null    object
```

```
 3   current_city       360 non-null    object
 4   current_education  360 non-null    object
 5   parent_occupation  360 non-null    object
 6   lead_gen_source    360 non-null    object
dtypes: int64(1), object(6)
memory usage: 19.8+ KB
```

In [6]:
```
l_b_d.head(5)
```

Out[6]:

	lead_id	age	gender	current_city	current_education	parent_occupation	lead_gen_source
0	USR1001	16	FEMALE	Hyderabad	Intermediate	Private Employee	social_media
1	USR1002	20	MALE	Bengaluru	B.Tech	Business	user_referrals
2	USR1003	20	FEMALE	Visakhapatnam	B.Tech	Lawyer	user_referrals
3	USR1004	16	MALE	Mumbai	Intermediate	IT Employee	user_referrals
4	USR1005	16	MALE	Chennai	Intermediate	Government Employee	user_referrals

In [7]:
```
l_b_d.tail(5)
```

Out[7]:

	lead_id	age	gender	current_city	current_education	parent_occupation	lead_gen_source
355	USR1356	21	MALE	Mumbai	Degree	Government Employee	user_referrals
356	USR1357	22	MALE	Chennai	Looking for Job	Government Employee	website
357	USR1358	25	MALE	Chennai	B.Tech	Government Employee	SEO
358	USR1359	18	FEMALE	Mumbai	B.Tech	Government Employee	email_marketing
359	USR1360	16	MALE	Mumbai	Intermediate	Government Employee	social_media

In [8]:
```
l_b_d.describe()
```

Out[8]:

	age
count	360.000000
mean	21.561111
std	11.555444
min	16.000000
25%	18.000000
50%	21.000000
75%	24.000000
max	211.000000

In [9]:
```
l_b_d.count()
```

```
lead_id              360
```

Out[9]:
```
age                 360
gender              360
current_city        360
current_education   360
parent_occupation   360
lead_gen_source     360
dtype: int64
```

In [10]:
```
l_b_d['age'].value_counts()
```

Out[10]:
```
18     68
20     64
24     50
22     47
25     47
21     42
16     40
211     1
116     1
Name: age, dtype: int64
```

Finding:

The leads_basic _details data shape shows the dataset has 360 rows & 7 columns.

Data info. shows the dataset has no null values.

Data of Age column shows the values(outliers) 116 & 211, which are >100 and hence required to exclude the both values.

In [11]:
```
print(l_b_d.loc[l_b_d['age']==211])

print(l_b_d.loc[l_b_d['age']==116])
```

```
    lead_id  age gender current_city current_education parent_occupation  \
17  USR1018  211   MALE    Hyderabad            Degree       IT Employee

   lead_gen_source
17    social_media
     lead_id  age  gender current_city current_education parent_occupation  \
300  USR1301  116  FEMALE    Hyderabad      Intermediate  Private Employee

    lead_gen_source
300    social_media
```

In [12]:
```
l_b_d.drop(index=[17,300],axis=0,inplace=True)
```

In [13]:
```
l_b_d['age'].value_counts()
```

Out[13]:
```
18    68
20    64
24    50
```

```
22    47
25    47
21    42
16    40
Name: age, dtype: int64
```

In [14]:
```
l_b_d
```

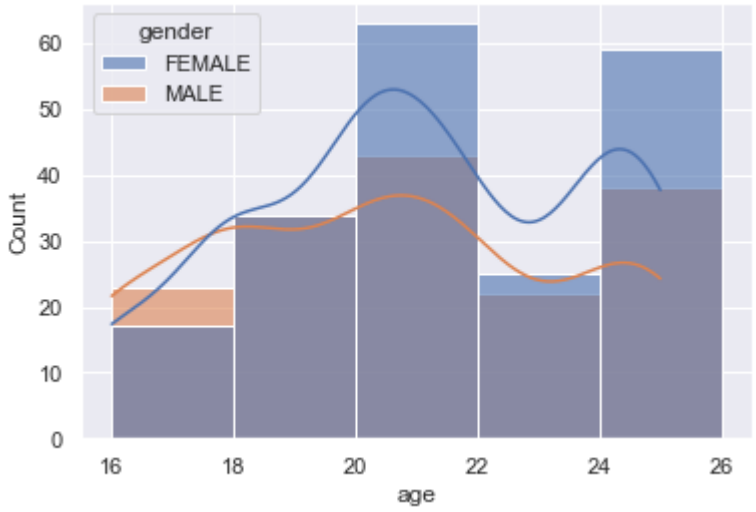
Out[14]:

	lead_id	age	gender	current_city	current_education	parent_occupation	lead_gen_source
0	USR1001	16	FEMALE	Hyderabad	Intermediate	Private Employee	social_media
1	USR1002	20	MALE	Bengaluru	B.Tech	Business	user_referrals
2	USR1003	20	FEMALE	Visakhapatnam	B.Tech	Lawyer	user_referrals
3	USR1004	16	MALE	Mumbai	Intermediate	IT Employee	user_referrals
4	USR1005	16	MALE	Chennai	Intermediate	Government Employee	user_referrals
...	...	...	...	...	...	...	...
355	USR1356	21	MALE	Mumbai	Degree	Government Employee	user_referrals
356	USR1357	22	MALE	Chennai	Looking for Job	Government Employee	website
357	USR1358	25	MALE	Chennai	B.Tech	Government Employee	SEO
358	USR1359	18	FEMALE	Mumbai	B.Tech	Government Employee	email_marketing
359	USR1360	16	MALE	Mumbai	Intermediate	Government Employee	social_media

358 rows × 7 columns

Let's check with the distribution of each detail from the above dataset individually

In [15]:
```
# Age
#sns.set(rc={'axes.facecolor':'#ddeedd', 'figure.facecolor':'#eeeeee'})
#sns.set()
sns.set(rc={'figure.figsize':(8,4)})
sns.set(rc={'figure.facecolor':'#ccddcc'})
sns.histplot(data=l_b_d, x='age', hue='gender',bins=9, binrange=(16,26), binwidth=2, al
sns.set(rc={'axes.facecolor':'#ffffff','figure.facecolor':'#ffffff'})
```



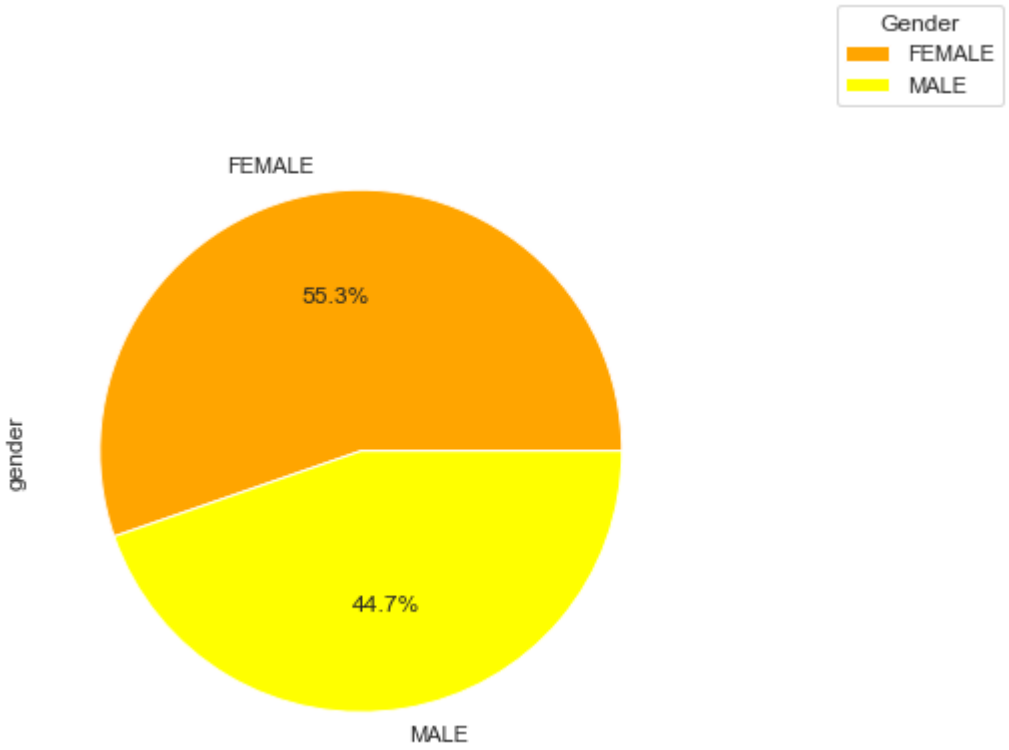
Finding:

Here we can observe the most leads are in between 20-22 and 24-26 age groups and the male ratio is greater than female in the age group of 16-18 and the rest majority are female from 18 -26 age group. Hence it is to be noted and anlayse the cause of low male proportion.

In [16]:
```
!pip install pip plotly
```

```
Requirement already satisfied: pip in c:\users\siddh\anaconda3\lib\site-packages (21.2.
4)
Requirement already satisfied: plotly in c:\users\siddh\anaconda3\lib\site-packages (5.1
0.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\siddh\anaconda3\lib\site-pack
ages (from plotly) (8.1.0)
```

In [17]:
```
# Gender
from chart_studio import plotly
import plotly.graph_objects as go

(l_b_d['gender'].value_counts()).plot(kind='pie',autopct='%1.1f%%', figsize=(6,6), colo
plt.legend(title ="Gender",loc ="best",bbox_to_anchor =(1, 0.2, 0.5, 1))
plt.show()
plt.tight_layout()
```




```
<Figure size 432x288 with 0 Axes>
```

In [18]:
```
import plotly.express as px
```

In [19]:
```
sns.set(rc={'figure.figsize':(12,4)})
with sns.axes_style(style=None):

    ax=sns.violinplot("current_city", "age", hue="gender", data=l_b_d,
                      split=True, inner="quartile",
                      palette=["violet", "lightblue"]);
    sns.move_legend(ax, "upper left", bbox_to_anchor=(1, 1))
```

```
C:\Users\siddh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

Finding: Female% > Male%

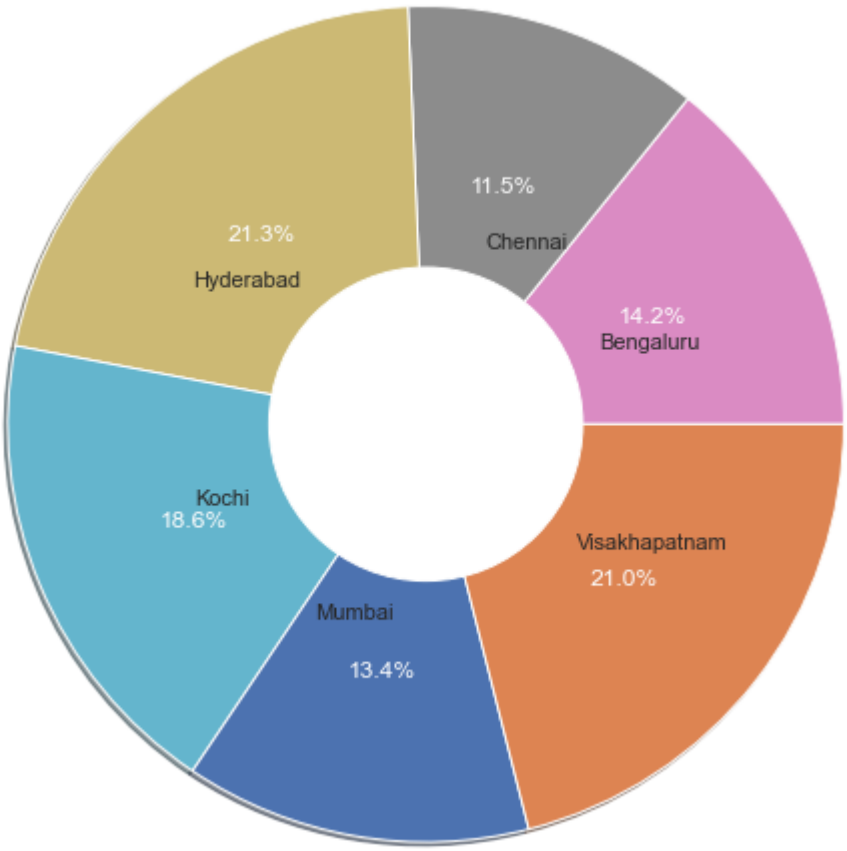
Anatomy of a violin chart:

--> The above violin chart summarises the dataset l_b_d using 6 measures showing the minimum, first
quartile, median, third quartile, and maximum and revealed the distribution

--> A wider PDF indicates that the value occurs more frequently, and a narrower density function
indicates that the value occurs less frequently.

--> Bengaluru, Mumbai & Chennai have comparitively narrow pdf at the initial stage but becomes
wider and reached to the high end however Kochi is narrower at both ends which is to be improved.

In [20]:
```
# Current_city
from numpy import array
dataset = ([52,42,78,68,49,77])
plt.pie(dataset, explode= [0,0,0,0,0,0], shadow= True, labels = ['Bengaluru', 'Chennai'
circle = plt.Circle( (0,0), 0.9, color='white')
plt.rcParams['text.color'] = 'white'
my_pie,_,_ = plt.pie(dataset,radius = 2.4,autopct="%.1f%%",shadow = True)
p=plt.gcf()
p.gca().add_artist(circle)
plt.show()
```

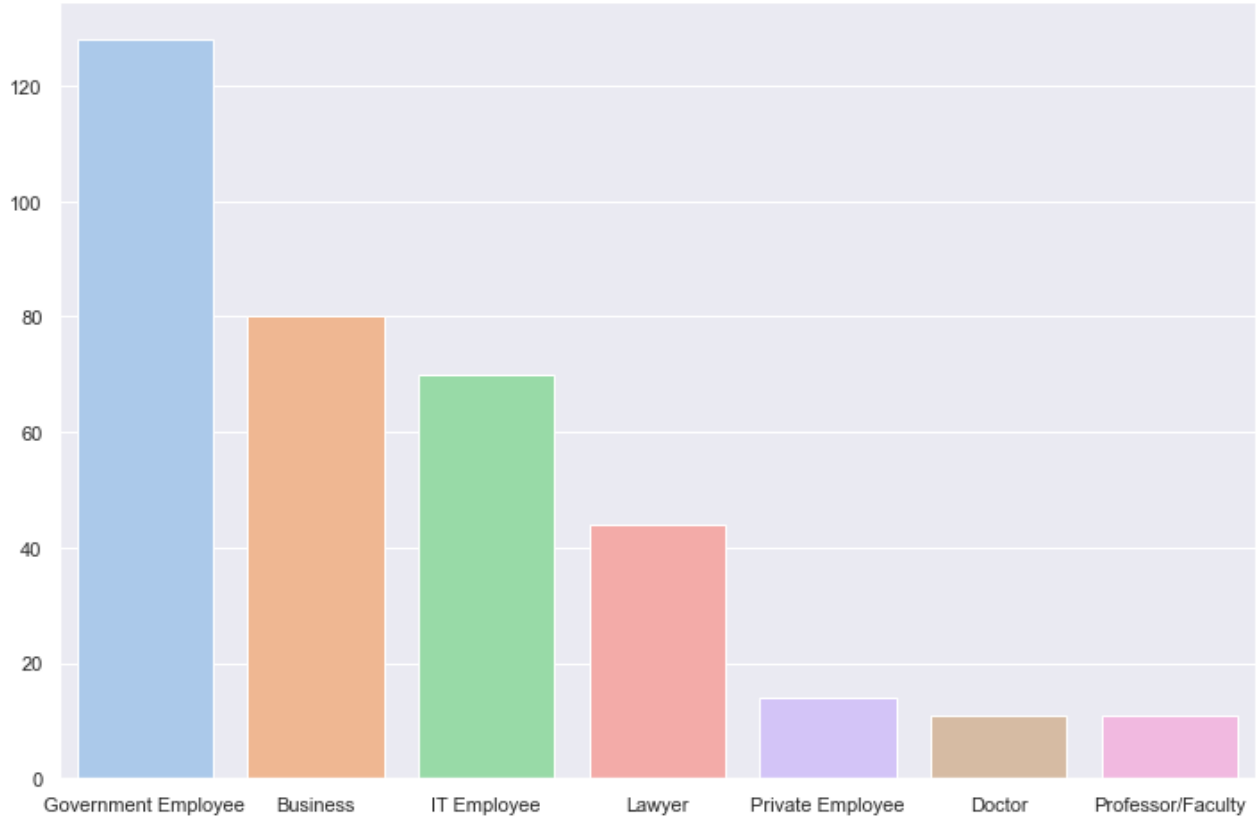


Finding:

--> From the above doughnut visual, we can see the majority of leads are high in proportion from the cities Hyderabad & Visakhapatnam where the scope of retention & acquisition could be maintained with by enhancing the services and offer incentives.

--> Chennai & Mumbai are in the least position and hence required attention in order to increase leads and can still be improved by offering attractive benefits in various aspects like enhancing program prospects, run quizzes/ surveys and can prompt for request referrals to improve customer acquisition.

In [21]:
```
# Parent
plt.figure(figsize=(12,8))
sns.barplot(x=l_b_d['parent_occupation'].value_counts().index,y=l_b_d['parent_occupatio
plt.show()
```

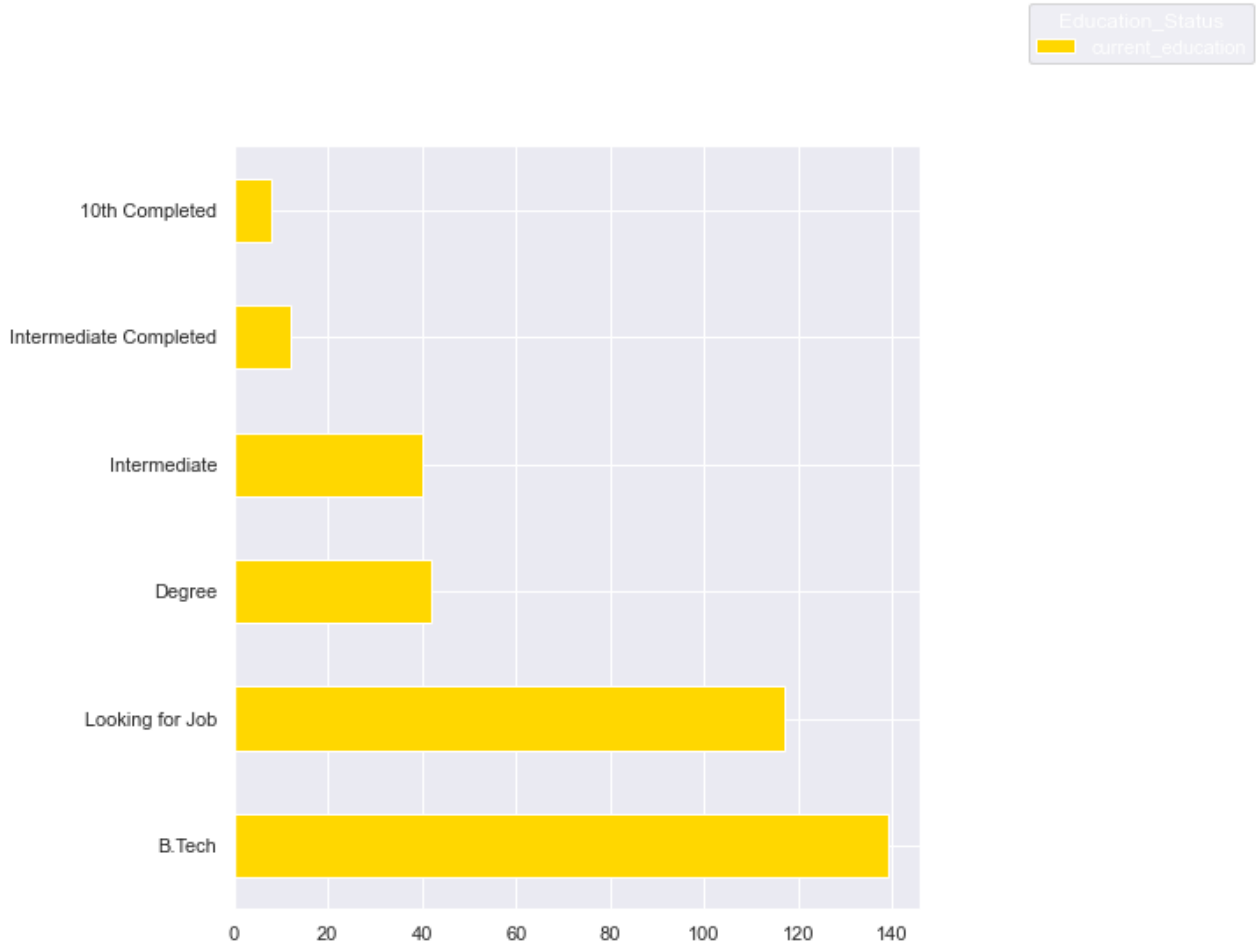


Finding:

--> The above bar plot depicts that the parents of the majority leads are from Government, IT & Business sectors.
--> Therefore it is important to attract and hold the leads even from the Private, Doctor & Professor/Faculty domain
and survey the reasons for the less number of leads from these particular parent group.

In [22]:
```
# Current_education
(l_b_d['current_education'].value_counts()).plot(kind='barh',figsize=(7,8), color = 'go
plt.legend(title ="Education_Status",loc ="best",bbox_to_anchor =(1, 0.2, 0.5, 1))
```

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




```
<Figure size 864x288 with 0 Axes>
```

Finding:

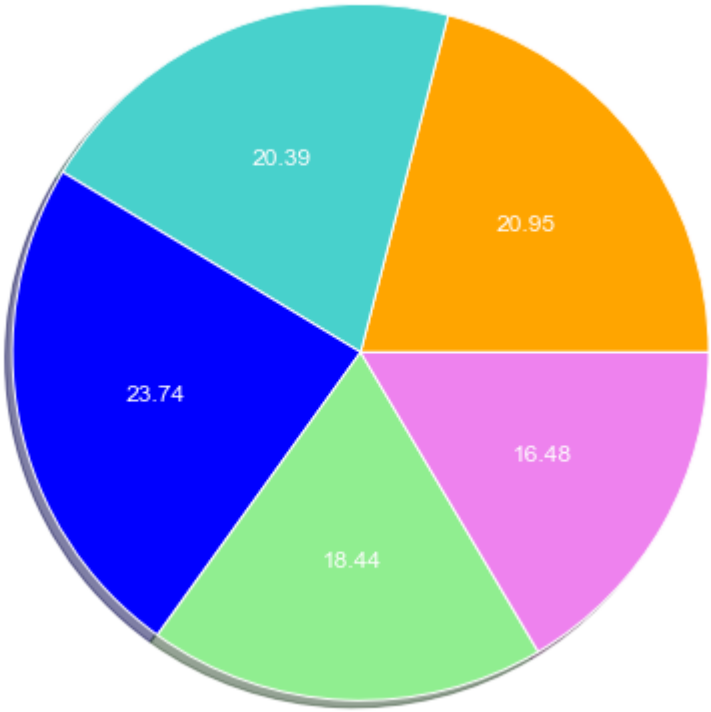
-- > Let's infer that the majority of the leads are from B.Tech and Looking for job categories. Here the other categories must be verified and enhance the program if required inorder to provide them feasibility and grab the leads as well.

In [23]:

```
# lead_gen_source
plt.figure(figsize=(15,8))
pie = l_b_d.groupby('lead_gen_source').size().plot(kind='pie', autopct='%.2f',  y='lead
pie.set_title("Lead_gen_Source")
```

Out[23]:

```
Text(0.5, 1.0, 'Lead_gen_Source')
```



Finding:

-- > The highest generating source is being contributed by Social Media(23.74), consecutively followed by SEO (20.95) & Email_marketing(20.39) , these sources can be used for new product offerings and campaign announcements.

-- > User_referrals & website sources are apparently low in generating leads and therefore these two referrals can be pinned and connected to Social_media and SEO where they will be provided with customer experience, review & rating which help them to choose the product/service.

Inference:

Through pre-processing the leads_basic_details it is intrepreted that ...

--> The most leads are in between 20-22 and 24-26 age groups.

--> The % of Female > % of Male.

--> The leads from the cities Hyderabad & Visakhapatnam are high in proportion.

--> The parents of the majority leads are from Government, IT & Business sectors.

--> The majority of the leads are from B.Tech and Looking for job.

--> The highest generating source is being contributed by Social Media(23.74), consecutively followed by SEO (20.95) & Email_marketing(20.39)

Dataset 2: leads_demo_watched_details

In [24]:
```
l_d_w_d.shape
```

Out[24]:
```
(194, 4)
```

In [25]:
```
l_d_w_d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194 entries, 0 to 193
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   lead_id             194 non-null    object
 1   demo_watched_date   194 non-null    object
 2   language            194 non-null    object
 3   watched_percentage  194 non-null    int64 
dtypes: int64(1), object(3)
memory usage: 6.2+ KB
```

In [26]:
```
l_d_w_d.head(5)
```

Out[26]:

	lead_id	demo_watched_date	language	watched_percentage
0	USR1002	1/4/2022	Telugu	42
1	USR1003	1/7/2022	Telugu	81
2	USR1004	1/2/2022	Telugu	35
3	USR1005	1/3/2022	Hindi	38
4	USR1006	1/12/2022	Hindi	54

In [27]:
```
l_d_w_d.tail(5)
```

Out[27]:

	lead_id	demo_watched_date	language	watched_percentage
189	USR1317	2/25/2022	English	48
190	USR1318	2/25/2022	English	83
191	USR1319	2/28/2022	English	84
192	USR1343	1/25/2022	English	68

	lead_id	demo_watched_date	language	watched_percentage
193	USR1348	2/27/2022	English	72

In [28]:
```
l_d_w_d.describe()
```

Out[28]:

	watched_percentage
count	194.000000
mean	56.634021
std	43.555635
min	2.000000
25%	35.000000
50%	55.500000
75%	75.750000
max	510.000000

In [29]:
```
l_d_w_d.count()
```

Out[29]:
```
lead_id               194
demo_watched_date     194
language              194
watched_percentage    194
dtype: int64
```

Finding:

The leads_demo_watched_details data shape shows the dataset has 194 Rows & 4 Columns

Data info. shows the dataset has no null values.

In [30]:
```
# let's check with the watched percentage and ensure it does not exceed 100

l_d_w_d.query('watched_percentage > 100')
```

Out[30]:

	lead_id	demo_watched_date	language	watched_percentage
94	USR1138	2/20/2022	English	510
133	USR1213	1/20/2022	Telugu	233

Action:

Since the watched_percentage is >100 for two leads (USR1138 & USR1213), let's exclude the both.

In [31]:
```
l_d_w_d
```

Out[31]:

	lead_id	demo_watched_date	language	watched_percentage
0	USR1002	1/4/2022	Telugu	42
1	USR1003	1/7/2022	Telugu	81
2	USR1004	1/2/2022	Telugu	35
3	USR1005	1/3/2022	Hindi	38
4	USR1006	1/12/2022	Hindi	54
...	...	...	...	...
189	USR1317	2/25/2022	English	48
190	USR1318	2/25/2022	English	83
191	USR1319	2/28/2022	English	84
192	USR1343	1/25/2022	English	68
193	USR1348	2/27/2022	English	72

194 rows × 4 columns

Now, let's merge leads_basic_details and leads_demo_watched_details since lead_id is the common detail in both the sheets

In [32]:
```
merged_leads = pd.merge(l_d_w_d, l_b_d, how='inner', on = 'lead_id')
```

In [33]:
```
merged_leads.head()
```

Out[33]:

	lead_id	demo_watched_date	language	watched_percentage	age	gender	current_city	current_c
0	USR1002	1/4/2022	Telugu	42	20	MALE	Bengaluru	
1	USR1003	1/7/2022	Telugu	81	20	FEMALE	Visakhapatnam	
2	USR1004	1/2/2022	Telugu	35	16	MALE	Mumbai	Int
3	USR1005	1/3/2022	Hindi	38	16	MALE	Chennai	Int
4	USR1006	1/12/2022	Hindi	54	16	MALE	Kochi	Int

In [34]:
```
merged_leads.describe()
```

Out[34]:

	watched_percentage	age

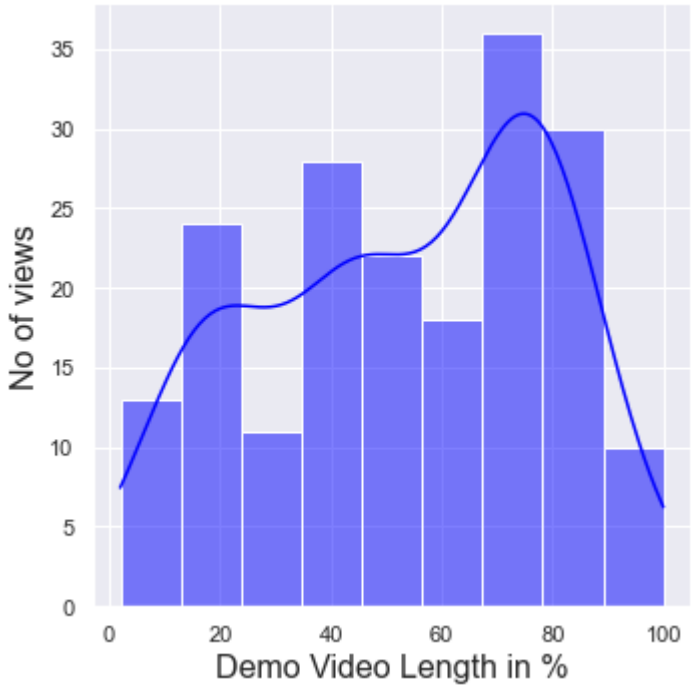
	watched_percentage	age
count	193.000000	193.000000
mean	56.720207	20.948187
std	43.652325	2.813178
min	2.000000	16.000000
25%	35.000000	18.000000
50%	56.000000	21.000000
75%	76.000000	24.000000
max	510.000000	25.000000

In [35]:
```
l_d_w_d = l_d_w_d.query('watched_percentage <= 100')
l_d_w_d.shape
```

Out[35]:
```
(192, 4)
```

In [36]:
```
sns.displot(l_d_w_d["watched_percentage"], kde=True, kind="hist", color='blue')
plt.title('View %', fontsize=16)
plt.xlabel('Demo Video Length in %', fontsize=16)
plt.ylabel('No of views', fontsize=16)
```

Out[36]:
```
Text(12.21, 0.5, 'No of views')
```



In [37]:
```
wp = merged_leads["watched_percentage"]
cc = merged_leads["current_city"]
dwd = merged_leads["demo_watched_date"]
```

```
fig = go.Figure(data=go.Heatmap(
        z=wp,
        x=dwd,
        y=cc,
        colorscale='Viridis'))
#Blackbody,Bluered,Blues,Cividis,Earth,Electric,Greens,Greys,Hot,Jet,Picnic,Portland,Ra

fig.update_layout(
    title='Watching Pattern',
    xaxis_nticks=365)

fig.show()
```

Finding:

-- > The insights provided by the above heat map about user interaction and behavior as they engage with the
demo product.

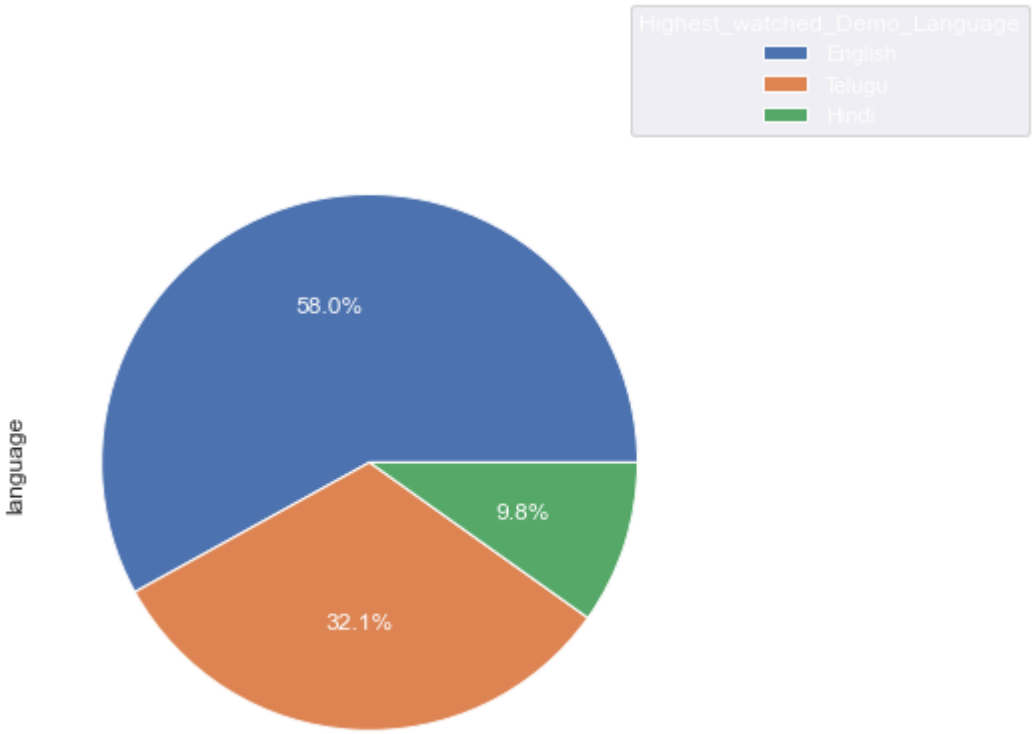
-- > In order to increase more views, and better conversion rates, the team has to focus on the less intensity regions
by determining what went well in the high density regions and its similarities & diffrences, types of user behaviour and to

penetrate and understand the psychology behind user clicks, scrolls, and choice.

In [38]:
```
merged_leads.groupby(['language', 'current_city']).size()
```

Out[38]:
```
language  current_city
English   Bengaluru        22
          Chennai          17
          Hyderabad        21
          Kochi            19
          Mumbai           11
          Visakhapatnam    22
Hindi     Bengaluru         3
          Chennai           6
          Hyderabad         3
          Kochi             4
          Mumbai            1
          Visakhapatnam     2
Telugu    Bengaluru         7
          Chennai           7
          Hyderabad        18
          Kochi             8
          Mumbai            7
          Visakhapatnam    15
dtype: int64
```

In [39]:
```
(merged_leads['language'].value_counts()).plot(kind='pie',autopct='%1.1f%%', figsize=(6
plt.legend(title ="Highest_watched_Demo_Language",loc ="best",bbox_to_anchor =(1, 0.2,
plt.show()
plt.tight_layout()
```



<Figure size 864x288 with 0 Axes>

Finding:

-- > The highest watched demo language is English with 58% and the least is Hindi with 9.8% and Telugu with 32.1% Try adding subtitles to Non-English demo videos for non-regional language leads & to research on the reasons for least view rate of Hindi demo video in the city like Mumbai and Bangalore.

Inference:

```
--> 100% video has been watched by 10% of leads and 50% of the video has
watched by the majority.
--> The most watched demo language is English with 58% and the least is
Hindi with 9.8% and Telugu with 32.1%
```

Dataset 3: leads_interaction_details Analysis

In [40]:
```
l_i_d.shape
```

Out[40]:
```
(2192, 6)
```

In [41]:
```
l_i_d.head()
```

Out[41]:

	jnr_sm_id	lead_id	lead_stage	call_done_date	call_status	call_reason
0	JNR1001MG	USR1001	lead	1/2/2022	successful	lead_introduction
1	JNR1001MG	USR1001	lead	1/2/2022	successful	demo_schedule
2	JNR1001MG	USR1002	lead	1/3/2022	successful	lead_introduction
3	JNR1001MG	USR1002	lead	1/4/2022	successful	demo_schedule
4	JNR1001MG	USR1002	awareness	1/5/2022	successful	post_demo_followup

In [42]:
```
# Let's merge l_d_w_d & l_i_d
merged_interaction_demo = pd.merge(l_d_w_d, l_i_d, how='inner', on = 'lead_id')
```

In [43]:
```
merged_interaction_demo.head()
```

Out[43]:

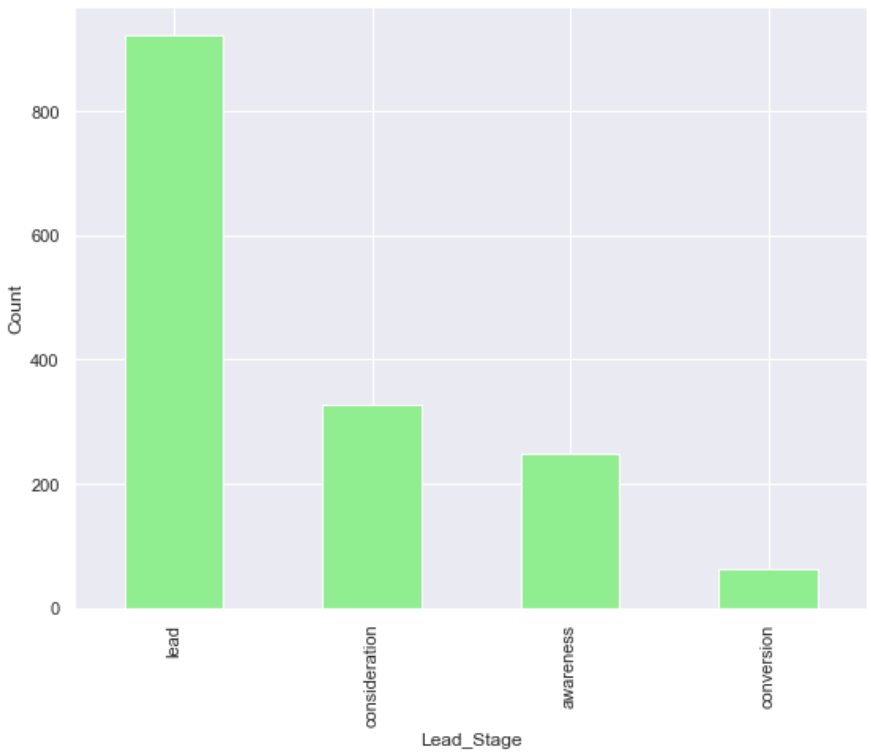
	lead_id	demo_watched_date	language	watched_percentage	jnr_sm_id	lead_stage	call_done_da
0	USR1002	1/4/2022	Telugu	42	JNR1001MG	lead	1/3/20
1	USR1002	1/4/2022	Telugu	42	JNR1001MG	lead	1/4/20
2	USR1002	1/4/2022	Telugu	42	JNR1001MG	awareness	1/5/20
3	USR1002	1/4/2022	Telugu	42	JNR1001MG	awareness	1/6/20
4	USR1002	1/4/2022	Telugu	42	JNR1001MG	consideration	1/7/20

In [44]:
```
merged_interaction_demo.describe()
```

Out[44]:

	watched_percentage
count	1560.000000
mean	53.848077
std	26.068039
min	2.000000
25%	30.000000
50%	60.000000
75%	76.000000
max	100.000000

In [45]:
```
(merged_interaction_demo['lead_stage'].value_counts()).plot(kind='bar',figsize=(9,7), c
plt.legend(title ="Viewed % of Lead Stage ",loc ="best",bbox_to_anchor =(1, 0.2, 0.5, 1
plt.xlabel("Lead_Stage")
plt.ylabel("Count")
plt.show()
plt.tight_layout()
```




```
<Figure size 864x288 with 0 Axes>
```

Inference:

-- > The proportion rate of conversion is too low when it's compared with lead & Consideration stage, hence it is required to focus on consideration & awareness stages as well.

-- > Seemingly 60% drop in the second stage itself (Consideration) that influences on the following stage which impacts on less coversion rate

Dataset 4: leads_reasons_for_no_interest

In [46]:
```
l_r_f_n_i.shape
```

Out[46]:
```
(294, 4)
```

In [47]:
```
l_r_f_n_i.head()
```

Out[47]:

	lead_id	reasons_for_not_interested_in_demo	reasons_for_not_interested_to_consider	reasons_for_not_i
0	USR1001	No time for student	NaN	
1	USR1003	NaN	No time for student	
2	USR1004	NaN	Wants offline classes	
3	USR1005	NaN	Can't afford	
4	USR1006	NaN	Student not interested in domain	

In [48]:
```
l_r_f_n_i.describe()
```

Out[48]:

	lead_id	reasons_for_not_interested_in_demo	reasons_for_not_interested_to_consider	reasons_for_
count	294	164	79	
unique	294	6	5	
top	USR1001	Wants offline classes	Can't afford	
freq	1	56	32	

Finding: From the above table it can be observed that there are 6 unique reasons for students not being interested in Demo Video and 5 reasons for not interested in consideration and convertion

In [49]:
```
l_r_f_n_i.groupby('reasons_for_not_interested_in_demo').size()
```

Out[49]:
```
reasons_for_not_interested_in_demo
Can't afford                         44
Cannot afford                         4
No time for student                  27
```

```
Student not interested in domain    28
Wants offline classes               56
Will join in final year              5
dtype: int64
```

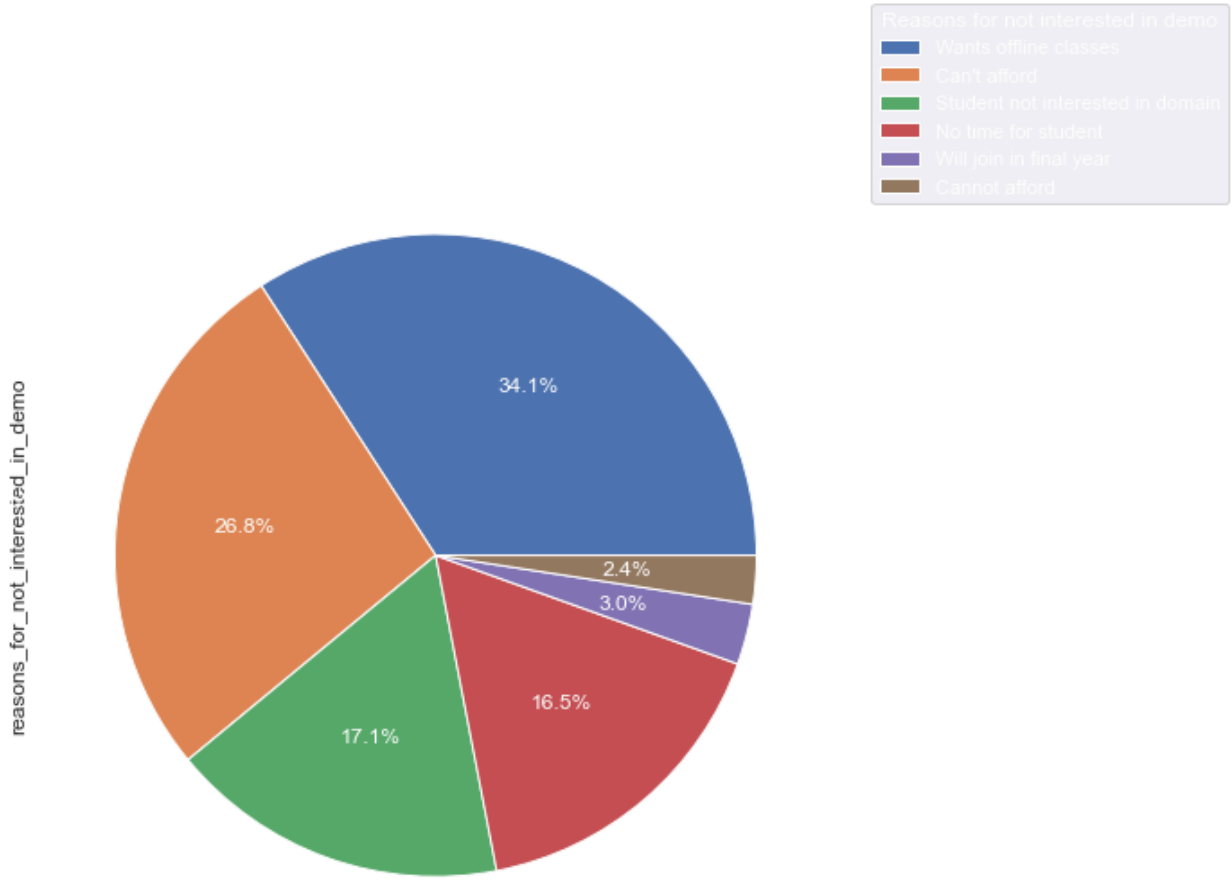
In [50]:
```
l_r_f_n_i
```

Out[50]:

	lead_id	reasons_for_not_interested_in_demo	reasons_for_not_interested_to_consider	reasons_for_not
0	USR1001	No time for student	NaN	
1	USR1003	NaN	No time for student	
2	USR1004	NaN	Wants offline classes	
3	USR1005	NaN	Can't afford	
4	USR1006	NaN	Student not interested in domain	
...	...	...	...	
289	USR1356	Cannot afford	NaN	
290	USR1357	Cannot afford	NaN	
291	USR1358	Wants offline classes	NaN	
292	USR1359	Will join in final year	NaN	
293	USR1360	Will join in final year	NaN	

294 rows × 4 columns

In [51]:
```
(l_r_f_n_i['reasons_for_not_interested_in_demo'].value_counts()).plot(kind='pie',autopc
plt.legend(title ="Reasons for not interested in demo",loc ="best",bbox_to_anchor =(1,
plt.show()
```



In [52]:
```
# Let's combine "Cannot afford" & "Can't afford by replacing either.

l_r_f_n_i['reasons_for_not_interested_in_demo'].replace({'Cannot afford':"Can't afford"
```

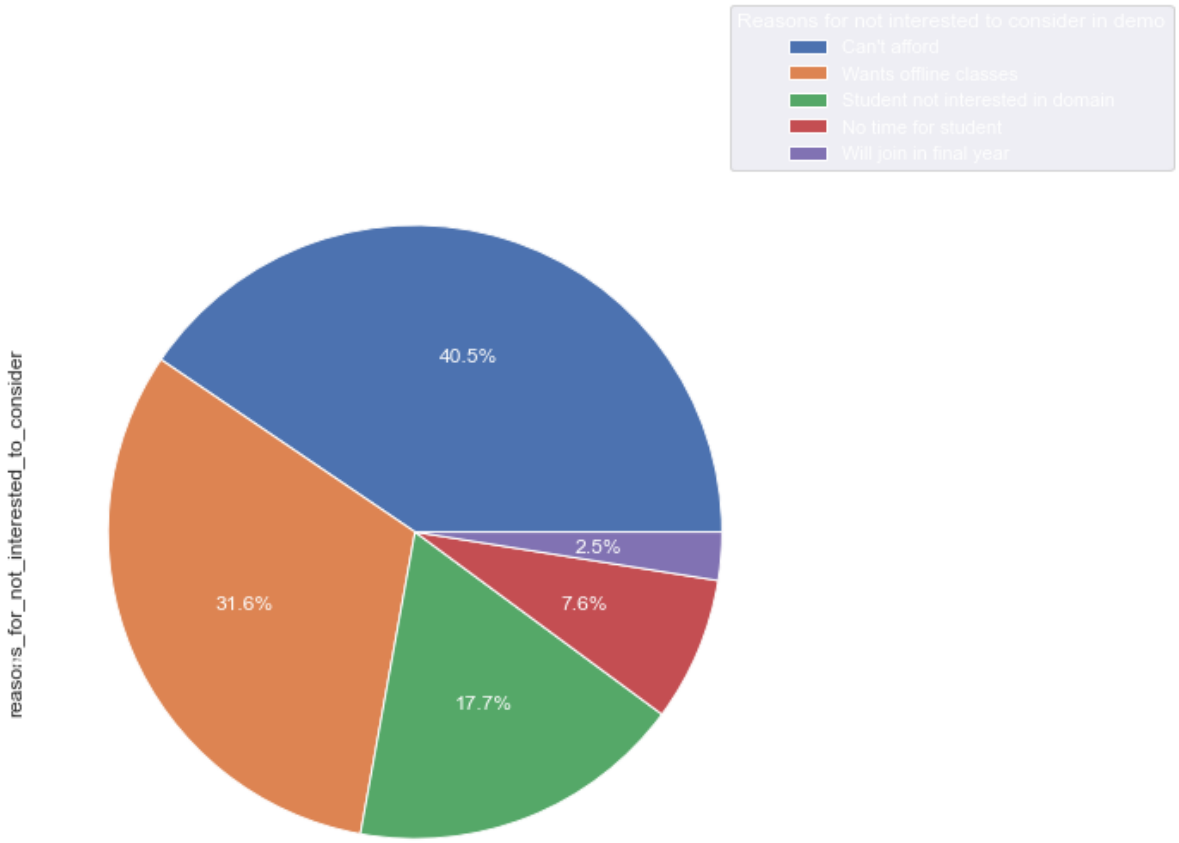
In [53]:
```
fig = px.scatter(x=["Wants offline classes", "Can't afford", 'Student not interested in
fig.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_21692/377078887.py in <module>
----> 1 fig = px.scatter(x=["Wants offline classes", "Can't afford", 'Student not inter
ested in domain', 'No time for student', 'Will join in final year'], y=[34.1, 29.2, 17.
1, 16.5, 3], size = 'reasons_for_not_interested_in_demo')
      2 fig.show()

~\anaconda3\lib\site-packages\plotly\express\_chart_types.py in scatter(data_frame, x,
 y, color, symbol, size, hover_name, hover_data, custom_data, text, facet_row, facet_co
l, facet_col_wrap, facet_row_spacing, facet_col_spacing, error_x, error_x_minus, error_
y, error_y_minus, animation_frame, animation_group, category_orders, labels, orientatio
n, color_discrete_sequence, color_discrete_map, color_continuous_scale, range_color, col
or_continuous_midpoint, symbol_sequence, symbol_map, opacity, size_max, marginal_x, marg
inal_y, trendline, trendline_options, trendline_color_override, trendline_scope, log_x,
 log_y, range_x, range_y, render_mode, title, template, width, height)
     64     mark in 2D space.
     65     """
---> 66     return make_figure(args=locals(), constructor=go.Scatter)
```

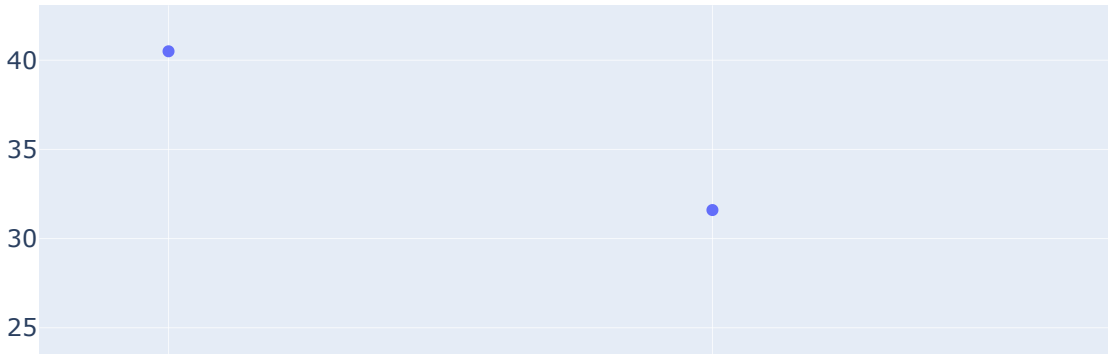
```
     67
     68

~\anaconda3\lib\site-packages\plotly\express\_core.py in make_figure(args, constructor,
 trace_patch, layout_patch)
   1988     apply_default_cascade(args)
   1989
-> 1990     args = build_dataframe(args, constructor)
   1991     if constructor in [go.Treemap, go.Sunburst, go.Icicle] and args["path"] is
not None:
   1992         args = process_dataframe_hierarchy(args)

~\anaconda3\lib\site-packages\plotly\express\_core.py in build_dataframe(args, construct
or)
   1403     # now that things have been prepped, we do the systematic rewriting of `args
`
   1404
-> 1405     df_output, wide_id_vars = process_args_into_dataframe(
   1406         args, wide_mode, var_name, value_name
   1407     )

~\anaconda3\lib\site-packages\plotly\express\_core.py in process_args_into_dataframe(arg
s, wide_mode, var_name, value_name)
   1187                     df_output[col_name] = to_unindexed_series(real_argument)
   1188                 elif not df_provided:
-> 1189                     raise ValueError(
   1190                         "String or int arguments are only possible when a "
   1191                         "DataFrame or an array is provided in the `data_frame` "

ValueError: String or int arguments are only possible when a DataFrame or an array is pr
ovided in the `data_frame` argument. No DataFrame was provided, but argument 'size' is o
f type str or int.
```

In [54]:

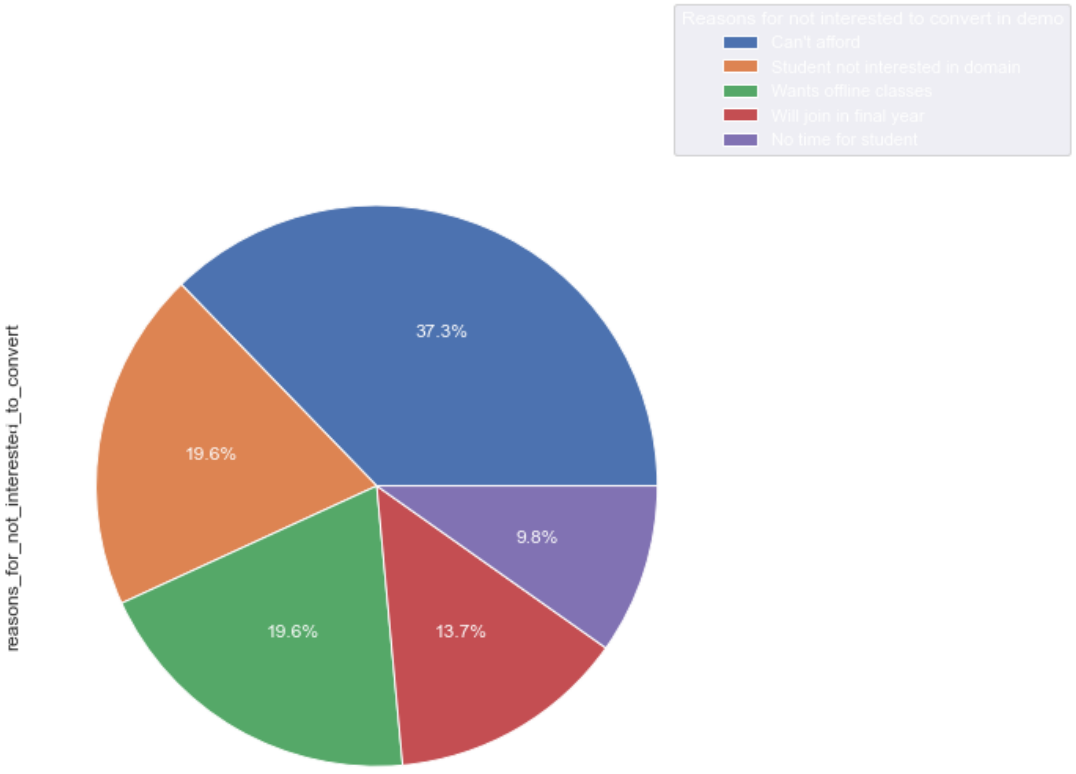
```
(l_r_f_n_i['reasons_for_not_interested_to_consider'].value_counts()).plot(kind='pie',au
plt.legend(title ="Reasons for not interested to consider in demo",loc ="best",bbox_to_
plt.show()
```



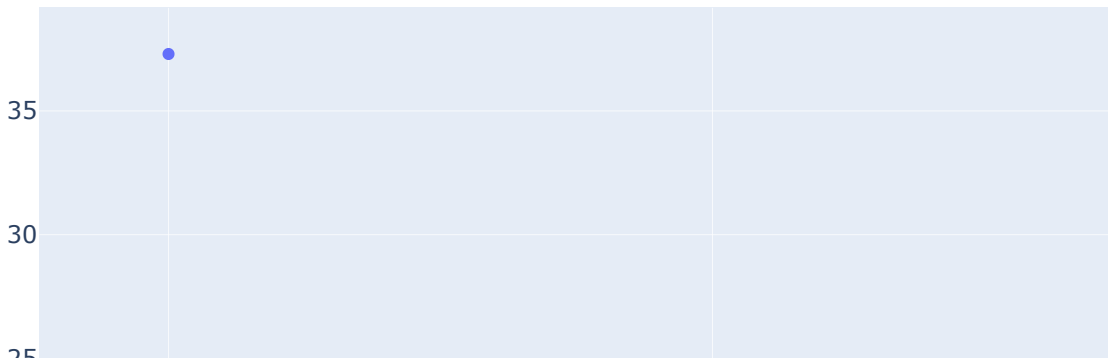
In [55]:
```
fig = px.scatter(x=["Can't afford", "Wants offline classes", "Student not interested in
fig.show()
```



In [56]:
```
(l_r_f_n_i['reasons_for_not_interested_to_convert'].value_counts()).plot(kind='pie',aut
plt.legend(title ="Reasons for not interested to convert in demo",loc ="best",bbox_to_a
plt.show()
```



In [57]:
```
fig = px.scatter(x=["Can't afford", "Student not interested in domain", "Wants offline
fig.show()
```



Inference:

-- > It's clearly evident that the prime reasons for leads dropping out at varied stages are affordability which can be analysed by two parameters, Parent_occupation & Current_education and the demand for offline classes.

-- > However the proportion of the leads who wanted offline classes & not interested in domain are equal.

-- > Hence these are the target points which are to be focussed and work on with an action plan.

Dataset 5: Sales_managers_assigned_leads_details

In [58]:
```
s_m_a_l_d.shape
```

Out[58]:
```
(360, 5)
```

In [59]:
```
s_m_a_l_d.head()
```

Out[59]:

	snr_sm_id	jnr_sm_id	assigned_date	cycle	lead_id
0	SNR501MG	JNR1001MG	1/1/2022	1	USR1001
1	SNR501MG	JNR1001MG	1/1/2022	1	USR1002
2	SNR501MG	JNR1001MG	1/1/2022	1	USR1003
3	SNR501MG	JNR1001MG	1/1/2022	1	USR1004
4	SNR501MG	JNR1001MG	1/1/2022	1	USR1005

In [60]:
```
merged_leads_reason = pd.merge(l_r_f_n_i, l_b_d, how='inner', on = 'lead_id')
```

In [61]:
```
merged_leads_reason
```

Out[61]:

	lead_id	reasons_for_not_interested_in_demo	reasons_for_not_interested_to_consider	reasons_for_not
0	USR1001	No time for student	NaN	
1	USR1003	NaN	No time for student	
2	USR1004	NaN	Wants offline classes	
3	USR1005	NaN	Can't afford	
4	USR1006	NaN	Student not interested in domain	
...	...	...	...	
287	USR1356	Can't afford	NaN	
288	USR1357	Can't afford	NaN	
289	USR1358	Wants offline classes	NaN	
290	USR1359	Will join in final year	NaN	
291	USR1360	Will join in final year	NaN	

292 rows × 10 columns

In [62]:
```
# Let's check with the reasons for the leads dropping out due to affordability which re

merged_leads_reason.query('reasons_for_not_interested_in_demo == ["Can\'t afford", "Can
```

Out[62]:
```
Looking for Job    26
B.Tech             16
Degree              3
Intermediate        2
10th Completed      1
Name: current_education, dtype: int64
```

In [63]:
```
merged_leads_reason.query('reasons_for_not_interested_to_consider == ["Can\'t afford",
```

Out[63]:
```
Looking for Job    16
Degree              6
Intermediate        4
B.Tech              3
10th Completed      2
Name: current_education, dtype: int64
```

In [64]:
```
merged_leads_reason.query('reasons_for_not_interested_to_convert == ["Can\'t afford", "
```

Out[64]:
```
Looking for Job    11
B.Tech              7
Degree              1
Name: current_education, dtype: int64
```

Finding: Since the majority of reasons are as the leads pursuing B.Tech/Degree or Looking for job, it is now required to check the parent_occupation

In [65]:
```
# Let's check with the reasons for the leads dropping out due to affordability which re

merged_leads_reason.query('reasons_for_not_interested_in_demo == ["Can\'t afford", "Can
```

Out[65]:
```
Government Employee    17
Business               11
IT Employee             9
Lawyer                  8
Private Employee        2
Doctor                  1
Name: parent_occupation, dtype: int64
```

In [66]:
```
merged_leads_reason.query('reasons_for_not_interested_to_consider == ["Can\'t afford",
```

Out[66]:
```
Government Employee    17
Business                6
IT Employee             4
Lawyer                  4
Name: parent_occupation, dtype: int64
```

In [67]:
```
merged_leads_reason.query('reasons_for_not_interested_to_convert == ["Can\'t afford", "
```

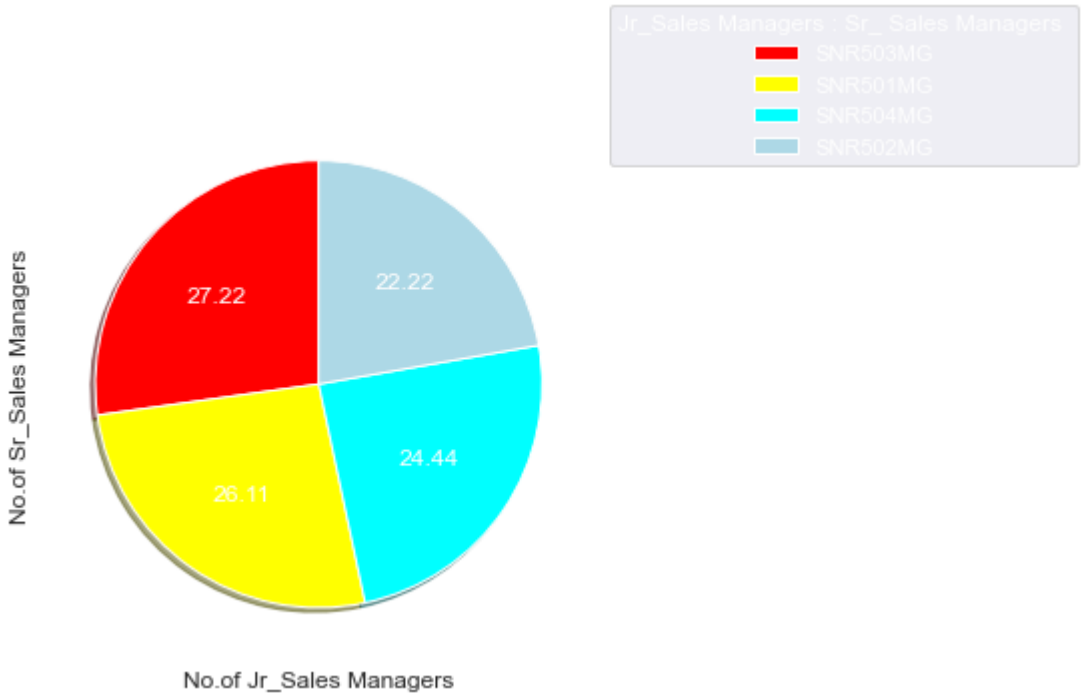
Out[67]:
```
Business               6
IT Employee            6
Government Employee    4
Lawyer                 2
Private Employee       1
Name: parent_occupation, dtype: int64
```

Finding:

-- > Most of the leads who dropped out at various stages due to affordability have parents working as a
government employee or an IT employee or in business.

-- > Since the majority of leads are from these parental groups, it is important to find the solution to
forbid droppings

In [68]:
```
s_m_a_l_d['snr_sm_id'].value_counts().plot(kind='pie',figsize=(5,8), autopct='%.2f',col
plt.legend(title ="Jr_Sales Managers : Sr_ Sales Managers ",loc ="best",bbox_to_anchor
plt.xlabel("No.of Jr_Sales Managers")
plt.ylabel("No.of Sr_Sales Managers")
plt.show()
plt.tight_layout()
```




```
<Figure size 864x288 with 0 Axes>
```

Finding: From the above pie-plot, it can be observed that the assignment of Junior Sales manager is merely close.

In [69]:
```
merged_interaction_demo.groupby([ 'lead_stage','call_status']).size()
```

Out[69]:
```
lead_stage     call_status
awareness      successful      243
               unsuccessful      5
consideration  successful      288
               unsuccessful     40
conversion     successful       63
lead           successful      769
               unsuccessful    152
dtype: int64
```

Finding: The amount of successful calls is greater than unsuccessful calls Conversion stage has 100% success rate

In [70]:
```
# Let's check the % of accomplished leads

accomplished_leads=l_i_d.query('call_reason == "successful_conversion"')['lead_id'].nun
accomplished_leads
```

Out[70]:
```
64
```

In [71]:
```
total_no_of_leads=l_i_d.lead_id.nunique()
total_no_of_leads
```

Out[71]:
```
358
```

In [72]:
```
no_of_leads_accomplished=np.array([accomplished_leads, total_no_of_leads-accomplished_l
```

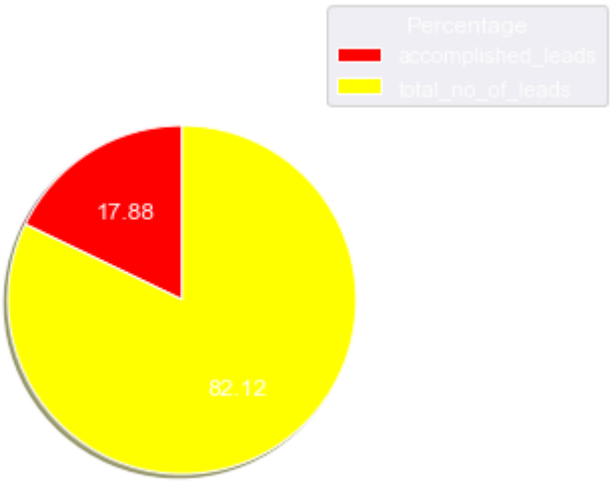
```
no_of_leads_accomplished
```

Out[72]:
```
array([ 64, 294])
```

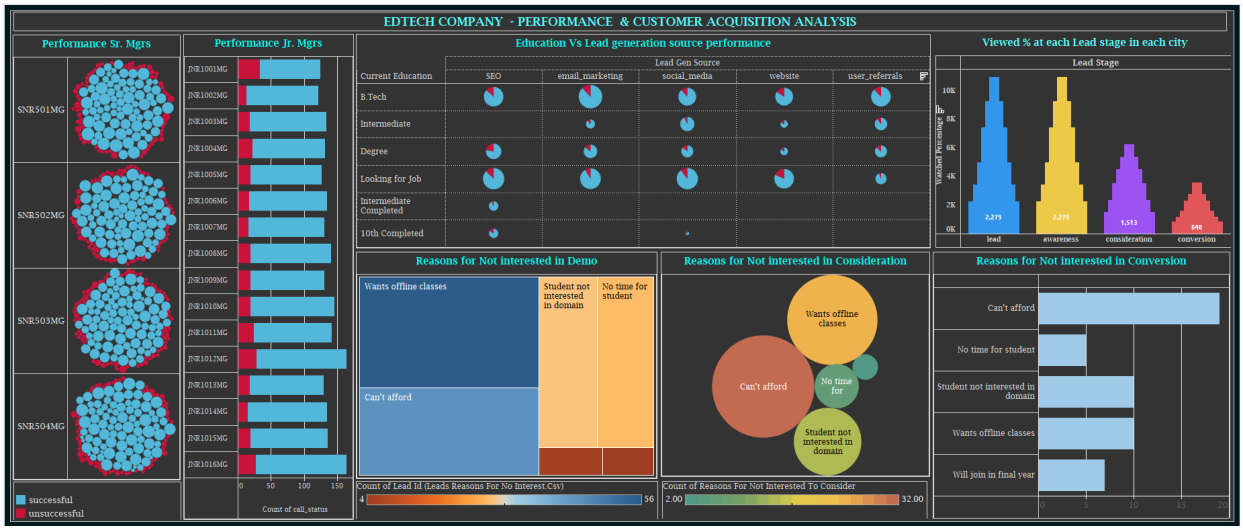
In [73]:
```
percent_of_leads_accomplished=accomplished_leads/total_no_of_leads*100
percent_of_leads_accomplished
```

Out[73]:
```
17.877094972067038
```

In [74]:
```
plt.pie(no_of_leads_accomplished,labels=["accomplished_leads","total_no_of_leads"],auto
plt.legend(title ="Percentage",loc ="best",bbox_to_anchor =(1, 0.2, 0.5, 1))
plt.show()
```



Inference: The total accomplishment of customer_acquisition by the team is 17.88%



In [ ]:
```

```