In [1]:

```
import pandas as pd

data = pd.read_excel('/content/drive/MyDrive/MockSurveyData.xlsx')
data.head(5)
```

Out[1]:

	case	Year	R.mth	date	intv_nam	Country_residence	City_residence	Purpose_grp	Purpose	Weights_QTR	Air_Terminal	Sea
0	18	2015	January	2015-01-01	Cindy Liu	Indonesia	Yogyakarta	Leisure	Holiday/ Rest & Relax	632.145161	Terminal 1	
1	41	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Leisure	Holiday/ Rest & Relax	341.937500	NaN	
2	43	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Leisure	Holiday/ Rest & Relax	432.866667	NaN	
3	44	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Leisure	Others	368.285714	NaN	
4	45	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Others/ Refused	Others - Personal (e.g. weddings, funerals, etc)	315.254902	NaN	

In [2]:

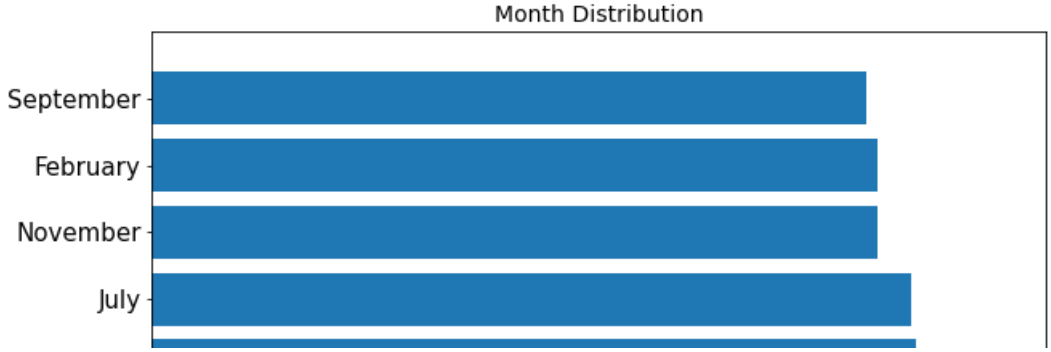
```
data['Year'].value_counts()
```

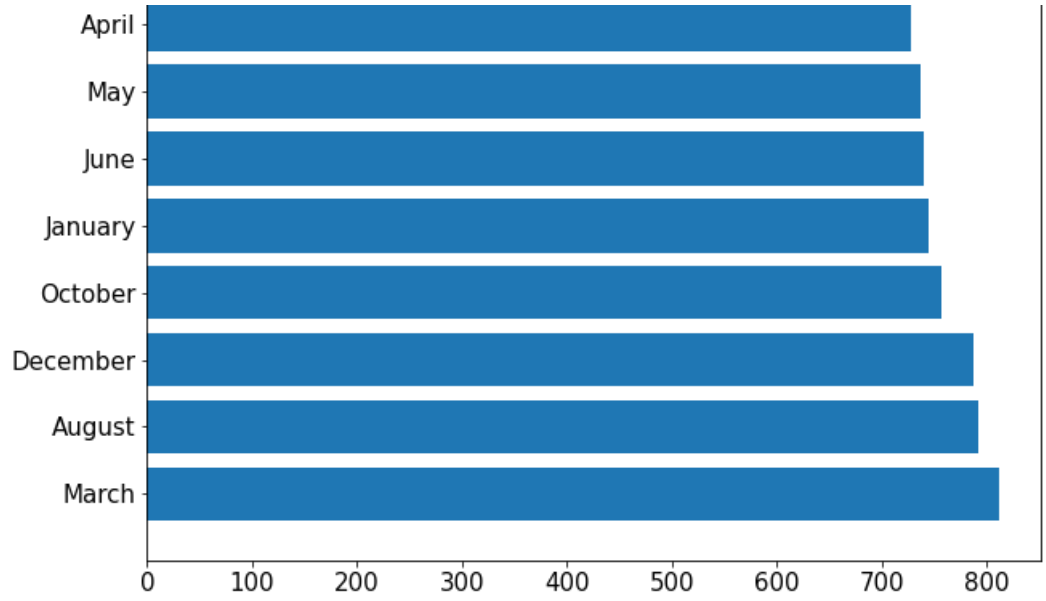
Out[2]:

```
2015    4475
2014    4412
Name: Year, dtype: int64
```

In [35]:

```
import matplotlib.pyplot as plt
month = dict(data['R.mth'].value_counts())

mth = list(month.keys())
count = list(month.values())

plt.figure(figsize= (10,10))
plt.barh(mth , count)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.title('Month Distribution', size = 14)
plt.show()
```

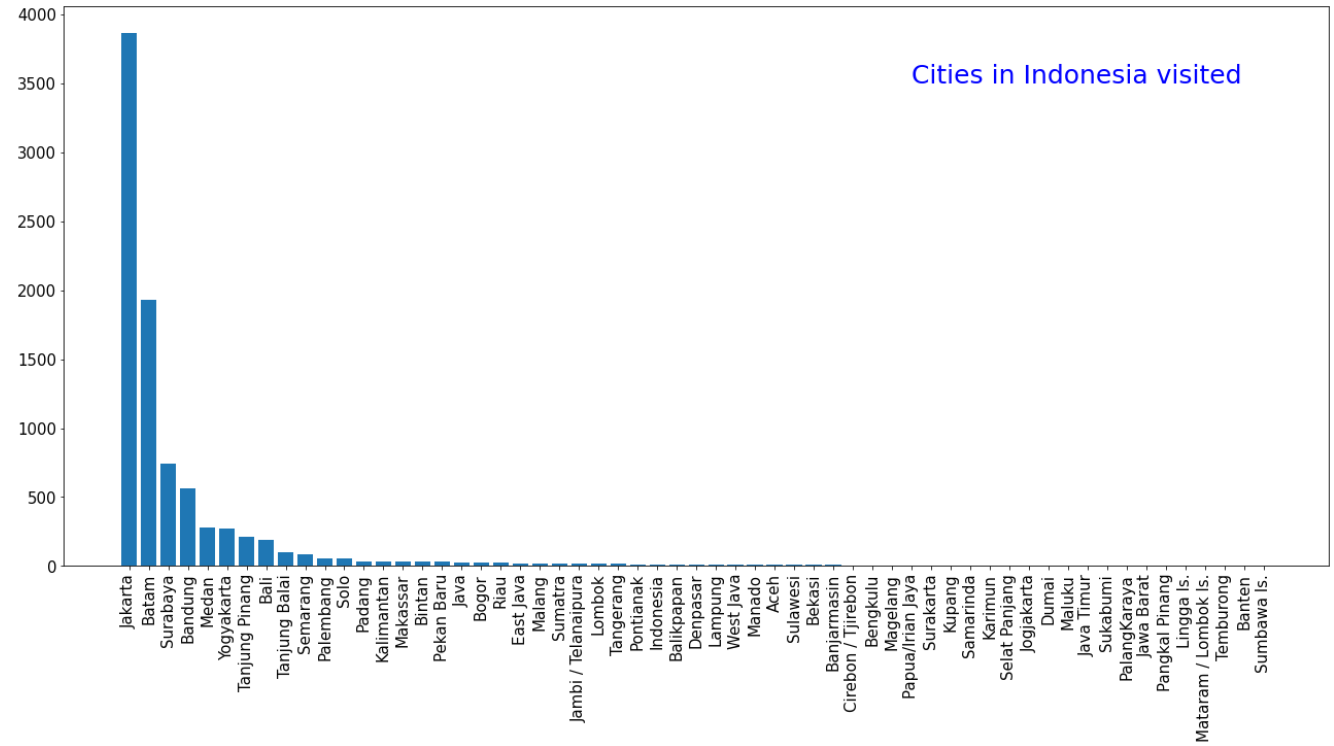




The above horizontal barplot shows that there is almost an equal significance of each month in the entire dataset

In [36]:

```
city = dict(data['City_residence'].value_counts())

import matplotlib.pyplot as plt
from matplotlib import text

category = list(city.keys())
values = list(city.values())

plt.figure(figsize= (22,10))
plt.bar(category , values)
plt.xticks(category, rotation='vertical', size = 15)
plt.text(40, 3500, 'Cities in Indonesia visited',size= 25 , color = 'blue')
plt.yticks(size = 15)
plt.show()
```



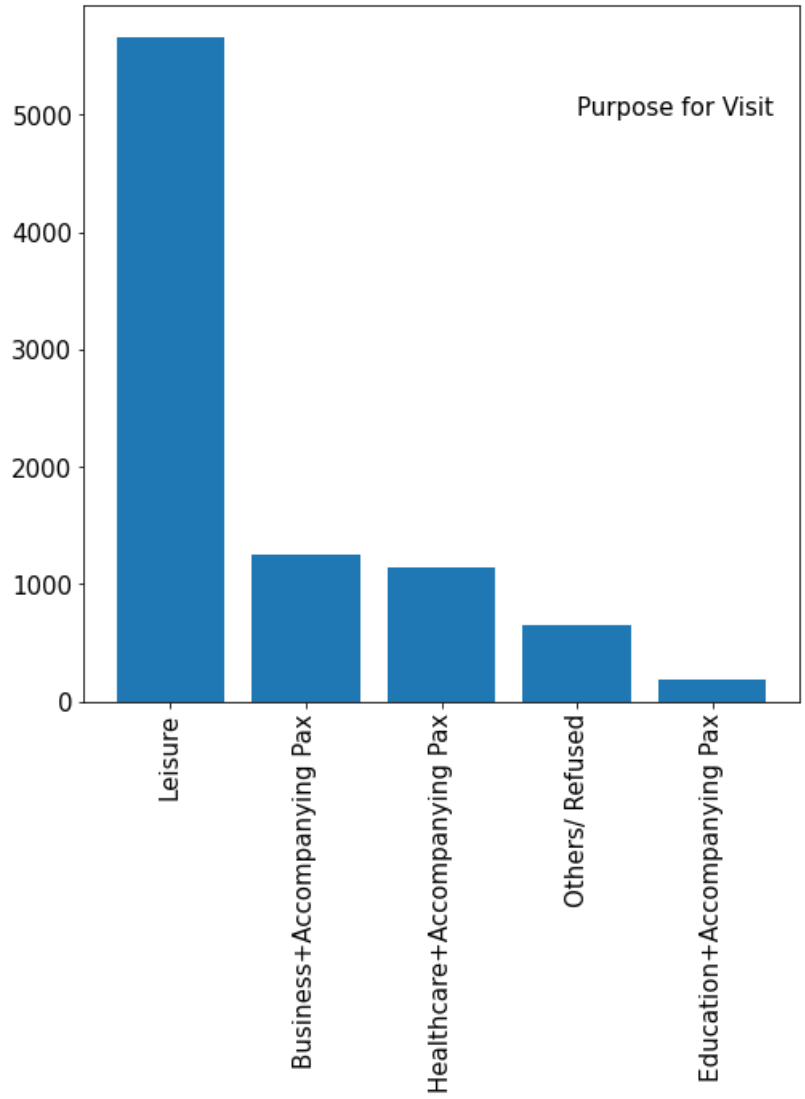
The above bar plot shows the most important areas to be kept in mind during data preparation are:

- Jakarta

- Jakarta
- Batam
- Surabaya
- Bandung
- Medan
- Yogyakarta
- Tanjung Pinang
- Bali
- Tanjung Balai
- Semarang
- Palembang
- Solo

In [37]:

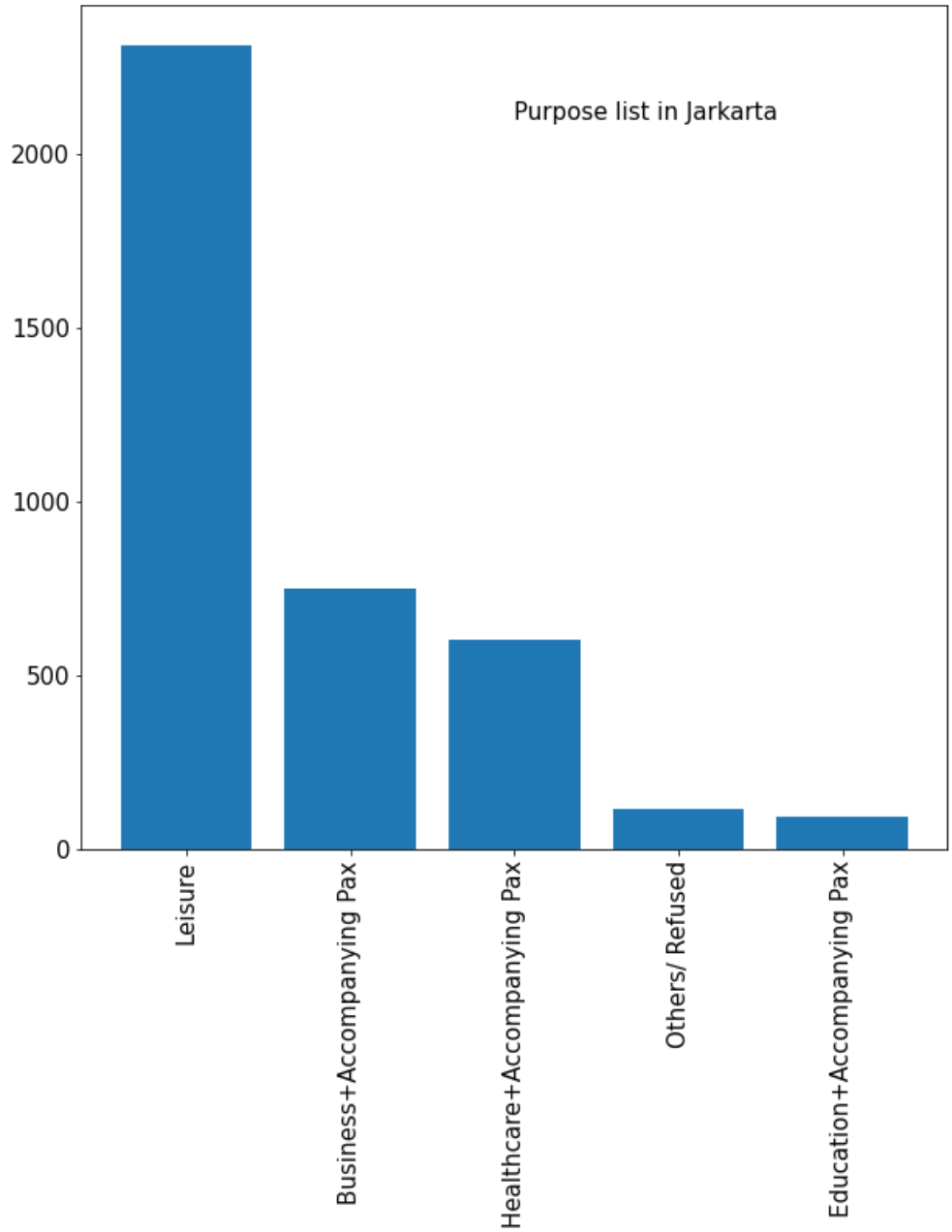
```
import matplotlib.pyplot as plt

purpose= dict(data['Purpose_grp'].value_counts())
category = list(purpose.keys())
values = list(purpose.values())

plt.figure(figsize= (8,8))
plt.bar(category , values)
plt.xticks(category, rotation= 90 , size = 15)
plt.yticks(size = 15)
plt.text(3, 5000 , 'Purpose for Visit ', size = 15)
plt.show()
```



Jakarta Analysis:

In [38]:

```
import matplotlib.pyplot as plt
data_jarkarta =  data[data['City_residence'] == 'Jakarta']

jakarta = dict(data_jarkarta['Purpose_grp'].value_counts())
category = list(jakarta.keys())
values = list(jakarta.values())

plt.figure(figsize= (10,10))
plt.bar(category , values)
plt.xticks(category, rotation= 90 , size = 15)
plt.yticks(size = 15)
plt.text(2, 2100 , 'Purpose list in Jarkarta', size = 15)
plt.show()
```

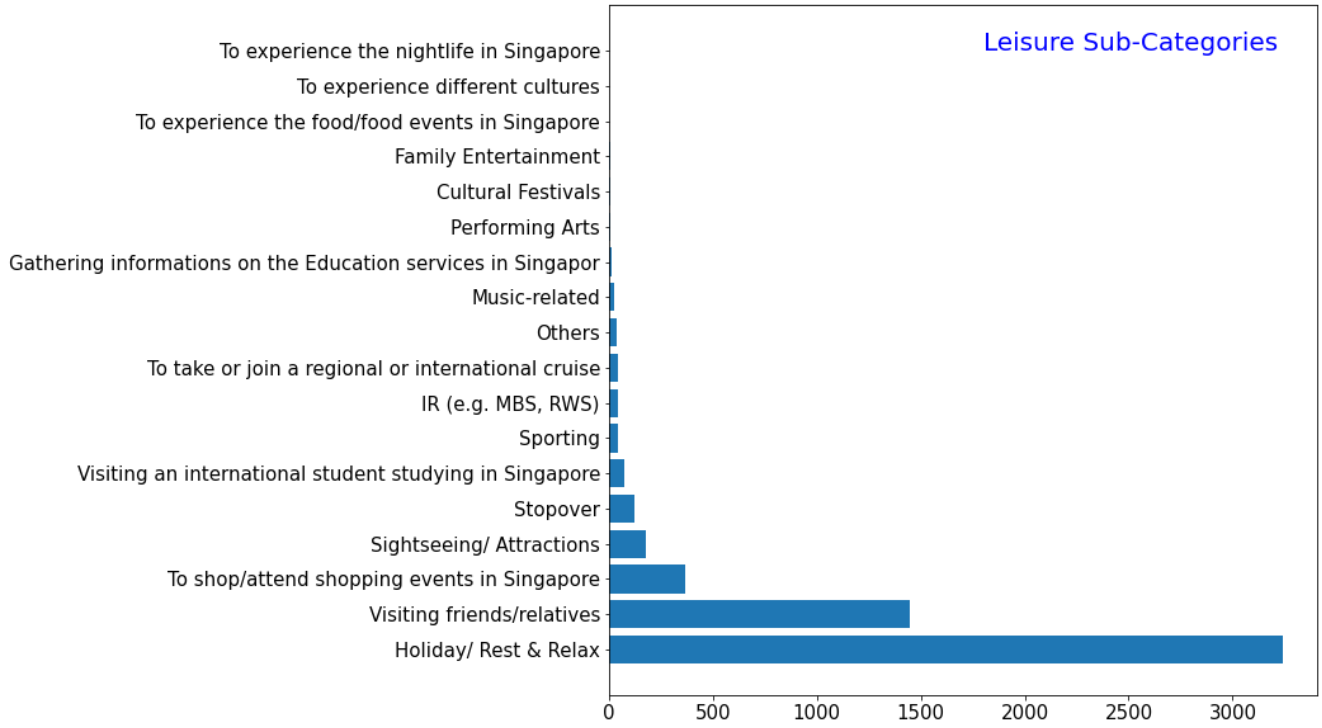


It seems most of the people visit jakarta for the purpose of Leisure

In [39]:

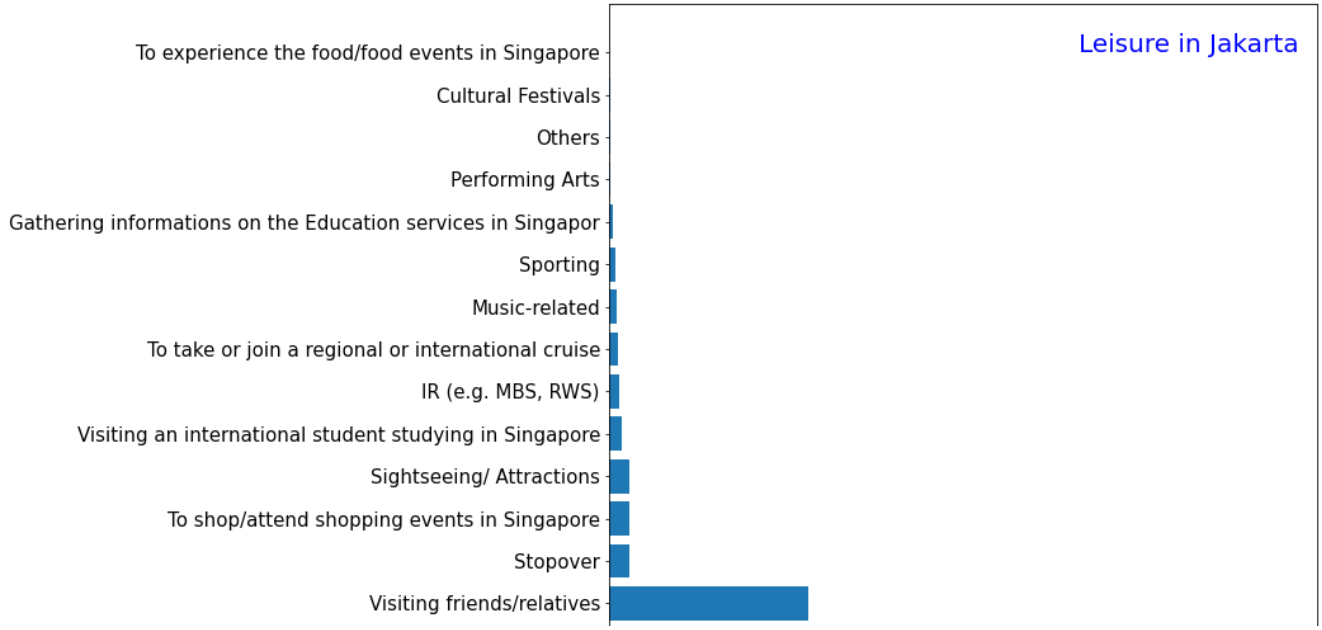
```
import matplotlib.pyplot as plt
data_leisure = data[data['Purpose_grp'] =='Leisure']

purpose = dict(data_leisure['Purpose'].value_counts())
category = list(purpose.keys())
values = list(purpose.values())
```

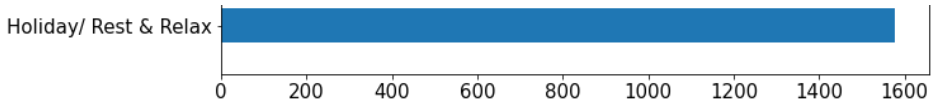
```
plt.figure(figsize= (10,10))
plt.barh(category , values)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.text(1800, 17 , 'Leisure Sub-Categories', size = 20 , color = 'blue')
plt.show()
```



In [40]:

```
data_jarkarta_purpose = data_jarkarta[data_jarkarta['Purpose_grp'] =='Leisure']
import matplotlib.pyplot as plt

purpose = dict(data_jarkarta_purpose['Purpose'].value_counts())
category = list(purpose.keys())
values = list(purpose.values())

plt.figure(figsize= (10,10))
plt.barh(category , values)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.text(1100, 14 , 'Leisure in Jakarta', size = 20 , color = 'blue')
plt.show()
```

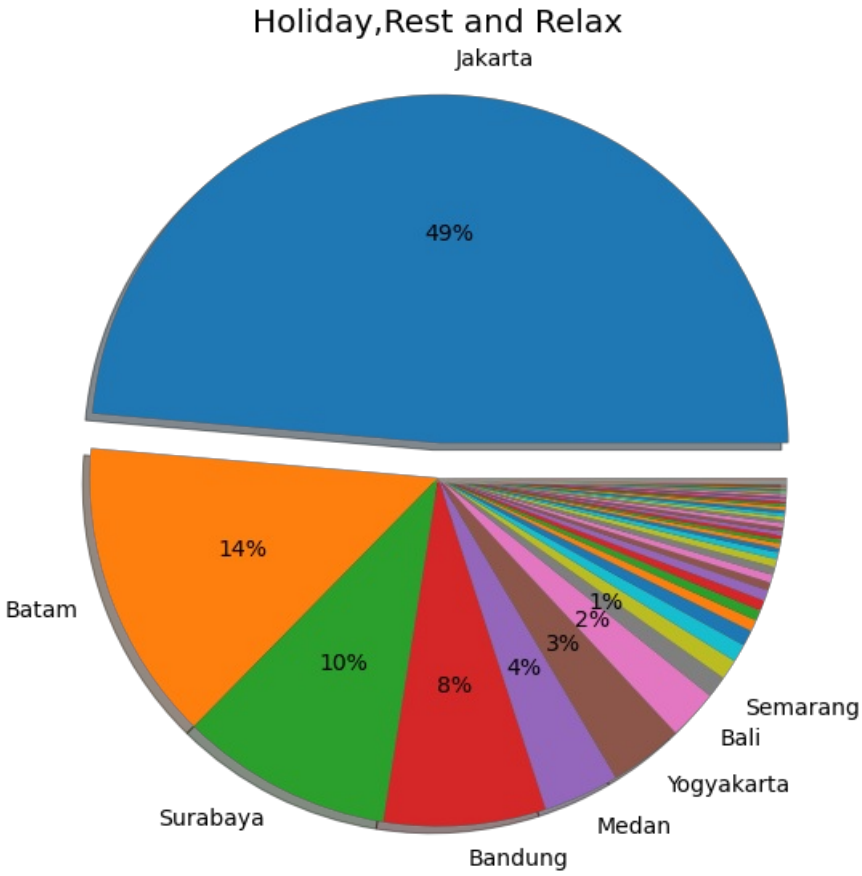




Here we can see that the top Category in Leisure Jarkarta is Holiday/Rest & Relax

In [41]:

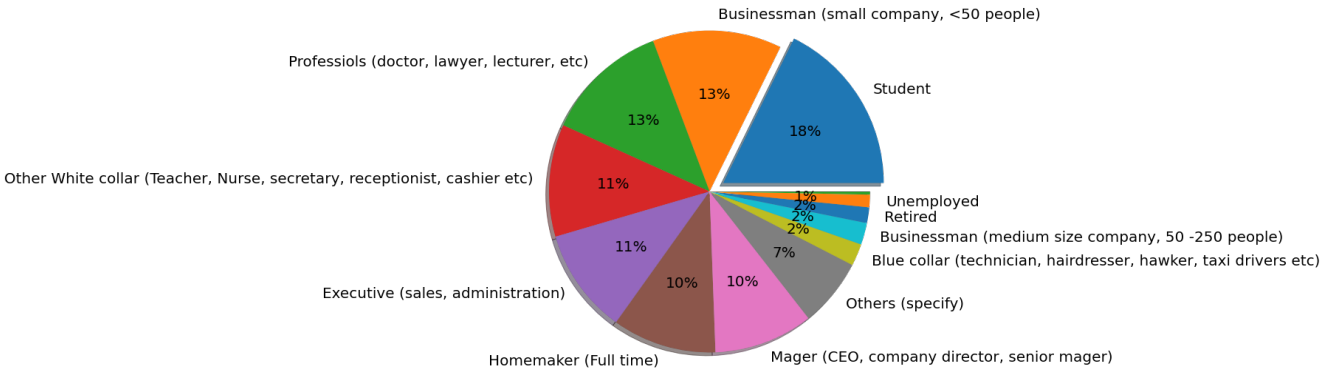
```
import matplotlib.pyplot as plt
import numpy as np

Holiday = data[data['Purpose']== 'Holiday/ Rest & Relax']
Holiday_city = dict(Holiday['City_residence'].value_counts())
category = list(Holiday_city.keys())
values = list(Holiday_city.values())

n = len(values)
explode = np.zeros(n)
explode[0] = 0.1
explode = tuple(explode)

label = []
for i in range(len(category)):
  if values[i] >= 31:
    label.append(category[i])
  elif values[i] < 31 :
    label.append('')

def autopct_more_than_1(pct):
    return ('%1.f%%' % pct) if pct > 1 else ''

plt.figure(figsize= (10,10))
plt.pie(values , explode = explode ,labels= label , autopct= autopct_more_than_1 ,textprops=dict(fo
ntsize = 14) , shadow=True)
plt.title('Holiday,Rest and Relax' , size = 20)
plt.show()
```



In the above pie chart we can see that the most significant city for Holiday , Rest and Realx as per the Dataset provided is Jakarta. Other main cities are Batam , Surabaya, Bandung , Medan , Yogyakarta and Bali which are significant for Holiday, Rest and Relax

In [42]:

```
import matplotlib.pyplot as plt
import numpy as np

Holiday = data[data['Purpose']== 'Holiday/ Rest & Relax']
Holiday_occupation = dict(Holiday['f3_occupation'].value_counts())
category = list(Holiday_occupation.keys())
values = list(Holiday_occupation.values())

n = len(values)
explode = np.zeros(n)
explode[0] = 0.1
explode = tuple(explode)

label = []
for i in range(len(category)):
  if values[i] >= 31:
    label.append(category[i])
  elif values[i] < 31 :
    label.append('')

def autopct_more_than_1(pct):
    return ('%1.f%%' % pct) if pct > 1 else ''

plt.figure(figsize= (10,10))
plt.pie(values , explode = explode ,labels= label , autopct= autopct_more_than_1 ,textprops=dict(fo
ntsize = 20) , shadow=True)
plt.show()
```



In [11]:

```
#Getting all the columns which have null values along with the number of null values that they hav
e in the entire column

data_columns = list(data.columns)
null_columns = []
non_null_columns = []
null_values = []
count = 0
for i in range(len(data_columns)):
  if data[data_columns[i]].isnull().sum() == 0:
    non_null_columns.append(data_columns[i])
    pass
  else:
    print(data_columns[i], end = " ")
    print('Column/Feature total number of NAN Values:', end= ' ')
    print(data[data_columns[i]].isnull().sum())
    count += 1
    null_columns.append(data_columns[i])
    null_values.append(data[data_columns[i]].isnull().sum())
```

```
Air_Terminal Column/Feature total number of NAN Values: 3190
Sea_Terminal Column/Feature total number of NAN Values: 6059
Land_Terminal Column/Feature total number of NAN Values: 8532
f4_industry Column/Feature total number of NAN Values: 2844
```

```
f5_designation Column/Feature total number of NAN Values: 3096
f5_designation.oth Column/Feature total number of NAN Values: 2969
shop_$fash Column/Feature total number of NAN Values: 80
shop_$jew Column/Feature total number of NAN Values: 80
shop_$wat Column/Feature total number of NAN Values: 80
shop_$well Column/Feature total number of NAN Values: 80
shop_$food Column/Feature total number of NAN Values: 80
shop_$gift Column/Feature total number of NAN Values: 80
shop_$ctec Column/Feature total number of NAN Values: 80
shop_$anti Column/Feature total number of NAN Values: 80
shop_$oth Column/Feature total number of NAN Values: 80
shop_$any Column/Feature total number of NAN Values: 80
MainAccomm Column/Feature total number of NAN Values: 33
MainHotel Column/Feature total number of NAN Values: 4590
travel_companion.2 Column/Feature total number of NAN Values: 7255
travel_companion.3 Column/Feature total number of NAN Values: 8397
travel_companion.4 Column/Feature total number of NAN Values: 8793
travel_companion.5 Column/Feature total number of NAN Values: 8870
```

In [12]:

```
data['length_stay'].value_counts()
```

Out[12]:

```
2 days            2159
Under 1 day       1827
1 day             1639
3 days            1460
4 days             699
5 days             335
6 days             199
15-29 days         174
8-10 days          156
7 days             134
11-14 days          75
30-59 days          29
60 days & over       1
Name: length_stay, dtype: int64
```

In [13]:

```
group = data.groupby(['length_stay','City_residence'])
group.first()
```

Out[13]:

		case	Year	R.mth	date	intv_nam	Country_residence	Purpose_grp	Purpose
length_stay	**City_residence**								
1 day	**Aceh**	12886	2015	August	2015-08-04	Anderson Yu Tor Kim	Indonesia	Leisure	Stopover
	Bali	847	2015	January	2015-01-10	Cindy Liu	Indonesia	Leisure	Holiday/ Rest & Relax
	Balikpapan	1621	2015	January	2015-01-20	Cindy Liu	Indonesia	Business+Accompanying Pax	General business purpose
	Bandung	550	2015	January	2015-01-06	Bay Poh Choo	Indonesia	Leisure	Visiting friends/relatives
	Banjarmasin	8698	2015	May	2015-05-14	Shariffah	Indonesia	Leisure	Holiday/ Rest & Relax
...	**...**	...	...	...	...	...	...	...	...
Under 1 day	**Surabaya**	1750	2015	January	2015-01-21	Teck Ghee	Indonesia	Healthcare+Accompanying Pax	Outpatient consultation/treatment

length_stay	City_residence	case	Year	R.mth	date	intv_nam	Country_residence	Purpose_grp	Purpose
	Tanjung Balai	864	2015	January	2015-01-10	Philip Chew	Indonesia	Leisure	Holiday/ Rest & Relax
	Tanjung Pinang	816	2015	January	2015-01-09	Teck Ghee	Indonesia	Others/ Refused	Others - Personal (e.g. weddings, funerals, etc)
	Temburong	10499	2014	June	2014-06-07	Philip Chew	Indonesia	Leisure	Holiday/ Rest & Relax
	Yogyakarta	866	2015	January	2015-01-10	Philip Chew	Indonesia	Others/ Refused	Others - Personal (e.g. weddings, funerals, etc)

349 rows × 46 columns

In [14]:

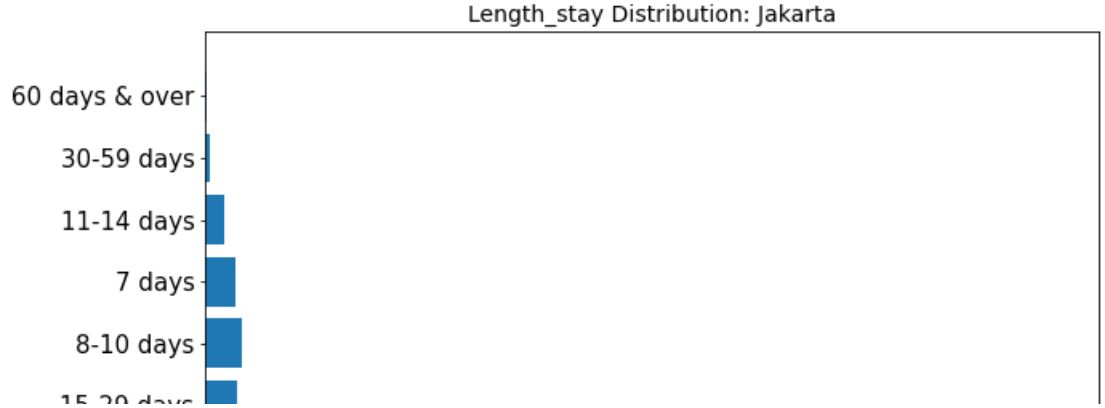
```
stay_len = dict(data['length_stay'].value_counts())
stay_len = list(stay_len.keys())

#print(stay_len)
jakarta_stay = {}
for i in range(len(stay_len)):
  jakarta_stay[stay_len[i]] = len(group.get_group((stay_len[i],'Jakarta' )))


#group.get_group(('1 day', 'Jakarta'))
jakarta_stay
```

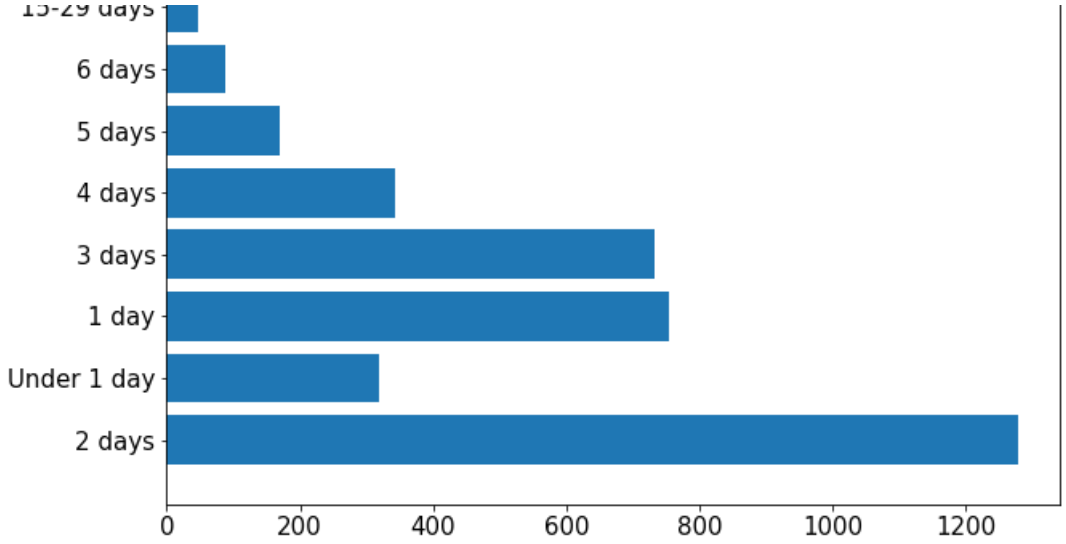
Out[14]:

```
{'1 day': 754,
 '11-14 days': 27,
 '15-29 days': 47,
 '2 days': 1279,
 '3 days': 733,
 '30-59 days': 7,
 '4 days': 344,
 '5 days': 169,
 '6 days': 89,
 '60 days & over': 1,
 '7 days': 45,
 '8-10 days': 54,
 'Under 1 day': 318}
```

In [43]:

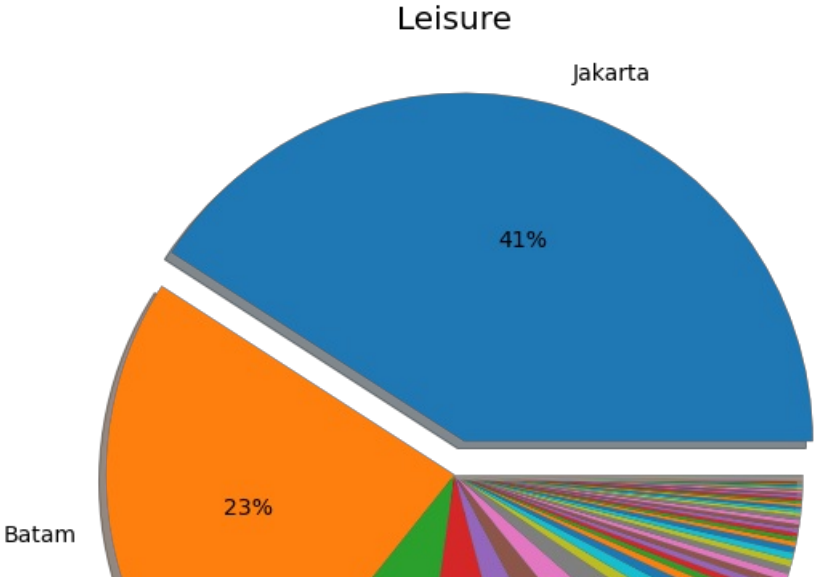
```
days = list(jakarta_stay.keys())
count = list(jakarta_stay.values())

plt.figure(figsize= (10,10))
plt.barh(days , count)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.title('Length_stay Distribution: Jakarta', size = 14)
plt.show()
```

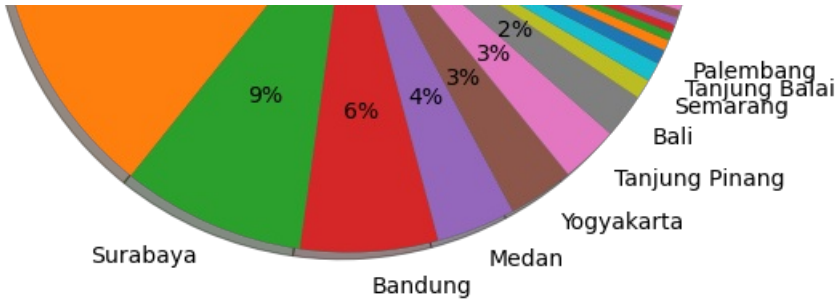




In [44]:

```
import matplotlib.pyplot as plt
import numpy as np

Purpose = data[data['Purpose_grp']== 'Leisure']
Purpose_city = dict(Purpose['City_residence'].value_counts())
#print(Purpose)
category = list(Purpose_city.keys())
values = list(Purpose_city.values())
#print(values)

n = len(values)
explode = np.zeros(n)
explode[0] = 0.1
explode = tuple(explode)

label = []
for i in range(len(category)):
  if values[i] >= 31:
    label.append(category[i])
  elif values[i] < 31 :
    label.append('')

def autopct_more_than_1(pct):
    return ('%1.f%%' % pct) if pct > 1 else ''

plt.figure(figsize= (10,10))
plt.pie(values , explode = explode ,labels= label , autopct= autopct_more_than_1 ,textprops=dict(fo
ntsize = 14) , shadow=True)
plt.title('Leisure' , size = 20)
plt.show()
```





In [17]:

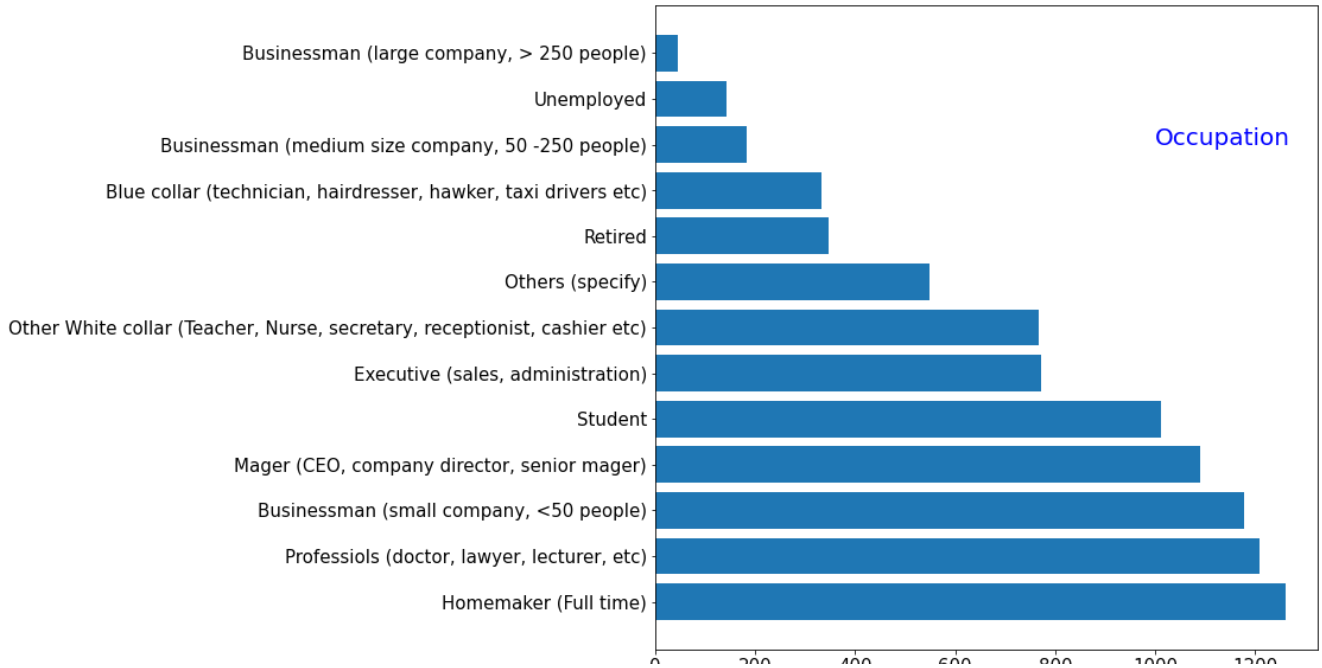
```
data['f3_occupation'].value_counts()
```

Out[17]:

```
Homemaker (Full time)                                                    1262
Professiols (doctor, lawyer, lecturer, etc)                              1208
Businessman (small company, <50 people)                                  1178
Mager (CEO, company director, senior mager)                              1090
Student                                                                  1012
Executive (sales, administration)                                         773
Other White collar (Teacher, Nurse, secretary, receptionist, cashier etc)  766
Others (specify)                                                          548
Retired                                                                   347
Blue collar (technician, hairdresser, hawker, taxi drivers etc)           332
Businessman (medium size company, 50 -250 people)                        183
Unemployed                                                                142
Businessman (large company, > 250 people)                                  46
Name: f3_occupation, dtype: int64
```

In [45]:

```
import matplotlib.pyplot as plt

occupation = dict(data['f3_occupation'].value_counts())
category = list(occupation.keys())
values = list(occupation.values())

plt.figure(figsize= (10,10))
plt.barh(category , values)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.text(1000, 10 , 'Occupation', size = 20 , color = 'blue')
plt.show()
```



0 200 400 600 800 1000 1200

In [19]:

```
occupation_group = data.groupby(['f3_occupation'])
occupation_group.first()
```

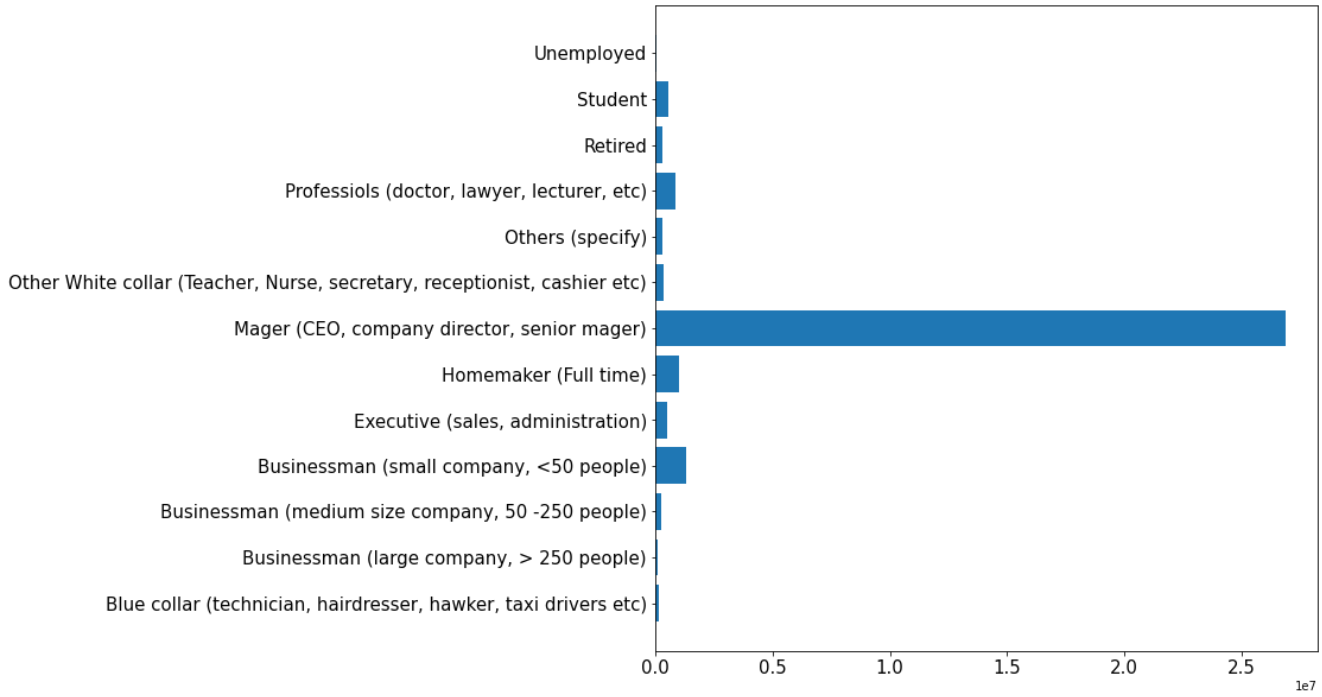
Out[19]:

f3_occupation	case	Year	R.mth	date	intv_nam	Country_residence	City_residence	Purpose_grp	Purpose	Weights_QTR
Blue collar (technician, hairdresser, hawker, taxi drivers etc)	366	2015	January	2015-01-04	Philip Chew	Indonesia	Batam	Leisure	Holiday/ Rest & Relax	368.285714
Businessman (large company, > 250 people)	1775	2015	January	2015-01-22	Anderson Yu Tor Kim	Indonesia	Jakarta	Leisure	Holiday/ Rest & Relax	310.762963
Businessman (medium size company, 50 - 250 people)	57	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Others/ Refused	Others - Personal (e.g. weddings, funerals, etc)	310.762963
Businessman (small company, <50 people)	18	2015	January	2015-01-01	Cindy Liu	Indonesia	Yogyakarta	Leisure	Holiday/ Rest & Relax	632.145161
Executive (sales, administration)	47	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Leisure	Holiday/ Rest & Relax	341.937500
Homemaker (Full time)	43	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Leisure	Holiday/ Rest & Relax	432.866667
Mager (CEO, company director, senior mager)	255	2015	January	2015-01-03	Simon	Indonesia	Jakarta	Leisure	Holiday/ Rest & Relax	419.090000
Other White collar (Teacher, Nurse, secretary, receptionist, cashier etc)	56	2015	January	2015-01-01	Philip Chew	Indonesia	Lampung	Leisure	Visiting friends/relatives	891.375000
Others (specify)	118	2015	January	2015-01-02	Cindy Liu	Indonesia	Surabaya	Leisure	Holiday/ Rest & Relax	351.881720
Professiols (doctor, lawyer, lecturer, etc)	41	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Leisure	Holiday/ Rest & Relax	341.937500
Retired	205	2015	January	2015-01-03	Philip Chew	Indonesia	Jakarta	Leisure	To take or join a regional or international cr...	1139.851852
Student	45	2015	January	2015-01-01	Philip Chew	Indonesia	Batam	Others/ Refused	Others - Personal (e.g. weddings, funerals, etc)	315.254902
Unemployed	555	2015	January	2015-01-06	Shariffah	Indonesia	East Java	Leisure	Visiting friends/relatives	621.195652

In [46]:

```
expenditure = dict(occupation_group.aggregate(np.sum )['totexp_$'])
category = list(expenditure.keys())
values = list(expenditure.values())

plt.figure(figsize= (10,10))
plt.barh(category , values)
```

```
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.show()

occupation_group.aggregate(np.sum )['totexp_$']
```



Out[46]:

```
f3_occupation
Blue collar (technician, hairdresser, hawker, taxi drivers etc)              1.627872e+05
Businessman (large company, > 250 people)                                   1.124812e+05
Businessman (medium size company, 50 -250 people)                           2.417534e+05
Businessman (small company, <50 people)                                     1.324163e+06
Executive (sales, administration)                                           5.043691e+05
Homemaker (Full time)                                                       1.032494e+06
Mager (CEO, company director, senior mager)                                 2.690093e+07
Other White collar (Teacher, Nurse, secretary, receptionist, cashier etc)   3.804690e+05
Others (specify)                                                            3.079584e+05
Professiols (doctor, lawyer, lecturer, etc)                                 8.519839e+05
Retired                                                                     3.273351e+05
Student                                                                     5.474782e+05
Unemployed                                                                  5.536989e+04
Name: totexp_$, dtype: float64
```

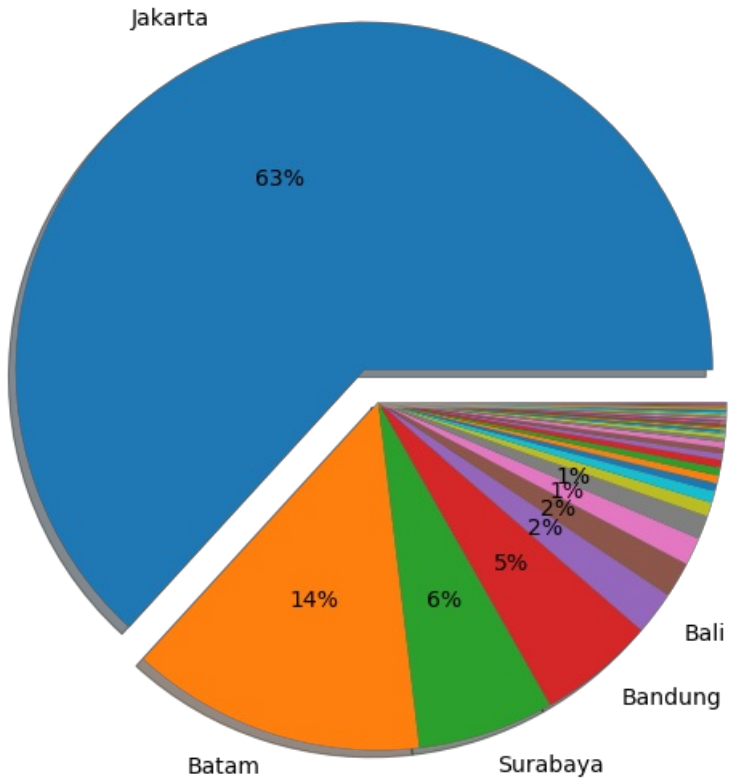
Maximum Expenditure is done by people belong to the Occupation of Mager (CEO, company director, senior mager)

Here the dataset indicates mager but it should be Manager I suppose

In [47]:

```
import matplotlib.pyplot as plt
import numpy as np

manager_data = data[data['f3_occupation']== 'Mager (CEO, company director, senior mager)']
manager_city = dict(manager_data['City_residence'].value_counts())
category = list(manager_city.keys())
values = list(manager_city.values())
#print(values)

n = len(values)
explode = np.zeros(n)
explode[0] = 0.1
explode = tuple(explode)

label = []
for i in range(len(category)):
  if values[i] >= 20:
```

```
    label.append(category[i])
  elif values[i] < 20 :
    label.append('')

def autopct_more_than_1(pct):
    return ('%1.f%%' % pct) if pct > 1 else ''

plt.figure(figsize= (10,10))
plt.pie(values , explode = explode ,labels= label , autopct= autopct_more_than_1 ,textprops=dict(fo
ntsize = 14) , shadow=True)
plt.show()
```



In [22]:

```
data.columns
```

Out[22]:

```
Index(['case', 'Year', 'R.mth', 'date', 'intv_nam', 'Country_residence',
       'City_residence', 'Purpose_grp', 'Purpose', 'Weights_QTR',
       'Air_Terminal', 'Sea_Terminal', 'Land_Terminal', 'langint', '1st_visit',
       'length_stay', 'travel_type', 'f1_gender', 'f3_occupation',
       'f4_industry', 'f5_designation', 'f5_designation.oth', 'shop_$fash',
       'shop_$jew', 'shop_$wat', 'shop_$well', 'shop_$food', 'shop_$gift',
       'shop_$ctec', 'shop_$anti', 'shop_$oth', 'shop_$any', 'totacc_$',
       'totfnb_$', 'tottran_$', 'totbiz_$', 'totedu_$', 'totmedi_$',
       'tototh_$', 'totshopping_$', 'totexp_$', 'MainAccomm', 'MainHotel',
       'travel_companion.1', 'travel_companion.2', 'travel_companion.3',
       'travel_companion.4', 'travel_companion.5'],
      dtype='object')
```

In [23]:

```
data['MainAccomm'].value_counts()
```

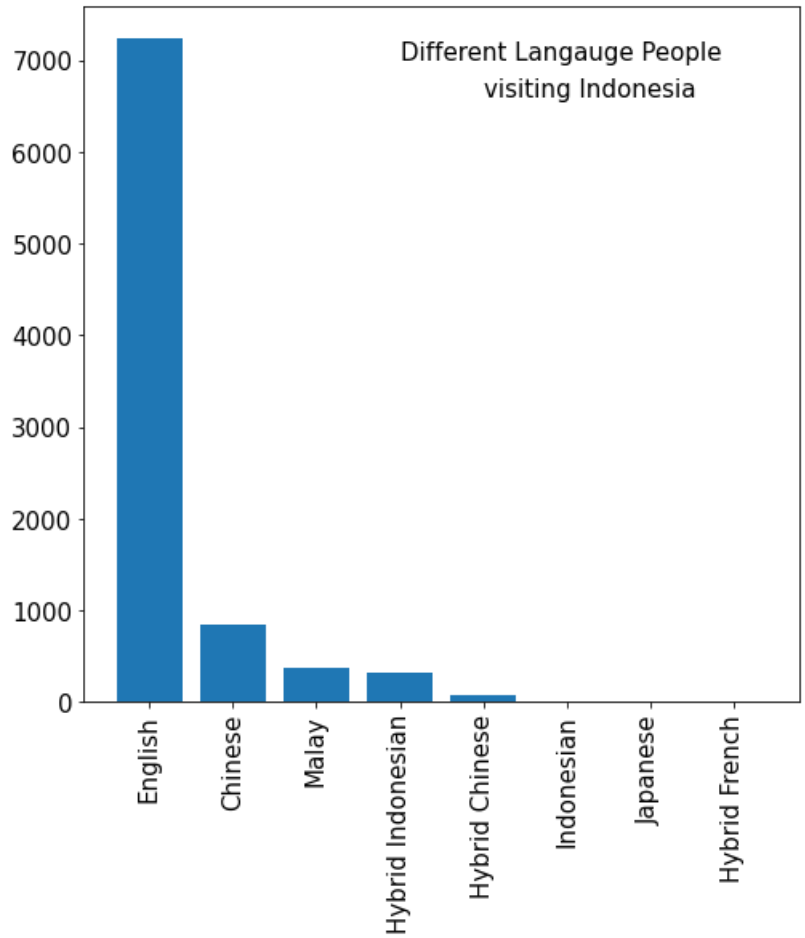
Out[23]:

```
Hotel                                                 4296
Stayed with relatives/ friends                        1811
Accommodation not required - Day Tripper              1746
Hostel (Rental by bed)                                 459
```

```
Hostel (Rental by Bed)                                              459
Service Apartment                                                   248
Own Residence                                                        89
Accommodation not required - On-board Cruise                         57
Homestay                                                             50
Other paid accommodations (e.g.chalets, country clubs, etc)          37
Accommodation not required - Others                                  28
Other non-paid accommodations (e.g. religious places, camp,          23
Student Hostel                                                        6
Hospital                                                              4
Name: MainAccomm, dtype: int64
```

In [48]:

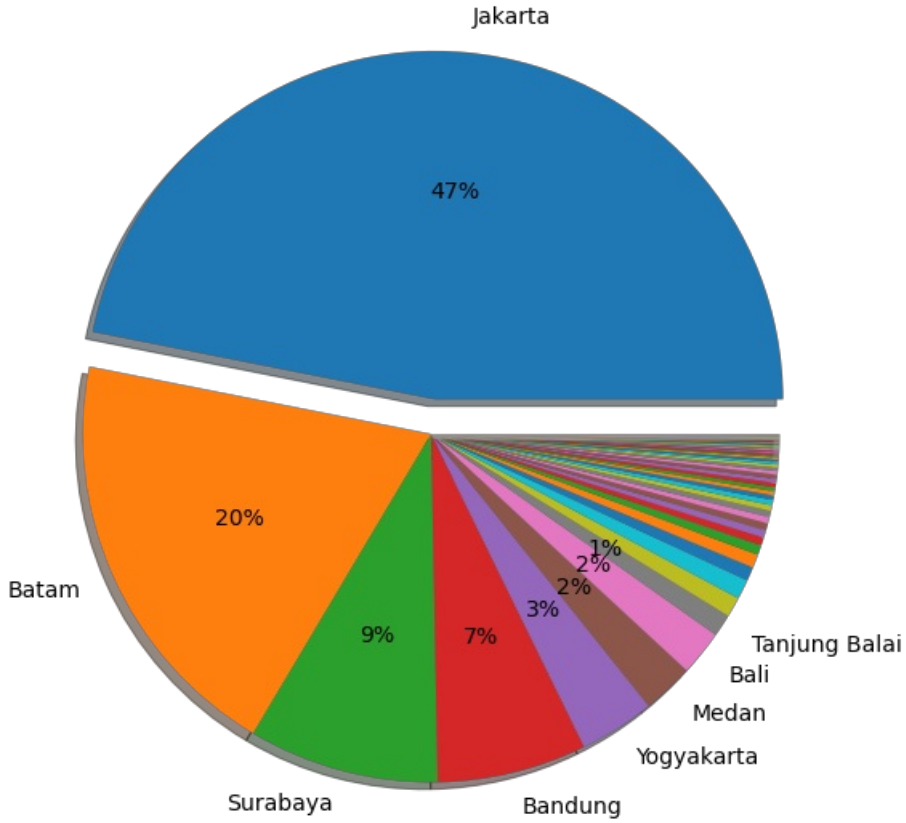
```
import matplotlib.pyplot as plt

language = dict(data['langint'].value_counts())
category = list(language.keys())
values = list(language.values())

plt.figure(figsize= (8,8))
plt.bar(category , values)
plt.xticks(category, rotation= 90 , size = 15)
plt.yticks(size = 15)
plt.text(3, 7000 , 'Different Langauge People' , size = 15)
plt.text(4, 6600 , 'visiting Indonesia' , size = 15)
plt.show()
```



In [49]:

```
import matplotlib.pyplot as plt
import numpy as np

language = data[data['langint']== 'English']
language_city = dict(language['City_residence'].value_counts())
#print(Purpose)
category = list(language_city.keys())
values = list(language_city.values())
#print(values)
```

```
n = len(values)
explode = np.zeros(n)
explode[0] = 0.1
explode = tuple(explode)

label = []
for i in range(len(category)):
  if values[i] >= 70:
    label.append(category[i])
  elif values[i] < 70 :
    label.append('')

def autopct_more_than_1(pct):
    return ('%1.f%%' % pct) if pct > 1 else ''

plt.figure(figsize= (10,10))
plt.pie(values , explode = explode ,labels= label , autopct= autopct_more_than_1 ,textprops=dict(fo
ntsize = 14) , shadow=True)
plt.show()
```

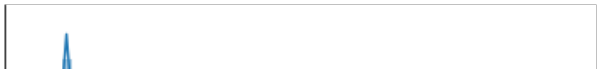


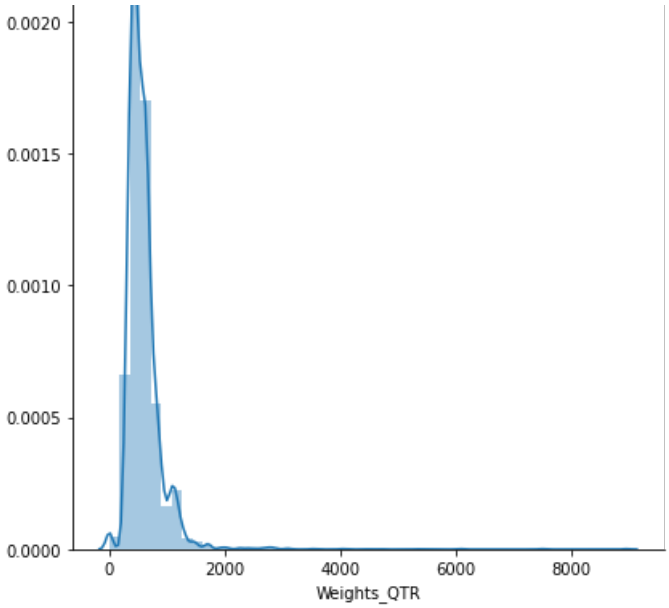
In [34]:

```
import seaborn as sns

plt.figure(figsize=(100,10))
sns.FacetGrid(data, height = 6).map(sns.distplot,'Weights_QTR');
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use eithe
r `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<Figure size 7200x720 with 0 Axes>
```




0.0020
0.0015
0.0010
0.0005
0.0000
0
2000
4000
6000
8000
Weights_QTR