```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
```

```
/kaggle/input/bankfullwithreqcol/bank-full.csv
```

```
Bankdf=pd.read_csv('../input/bankfullwithreqcol/bank-full.csv',delimiter=';')
Bankdf.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198

print(Bankdf.describe()) print(Bankdf.info())

```
Bankdf.isnull().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
```

```
previous     0
poutcome     0
y            0
dtype: int64
```

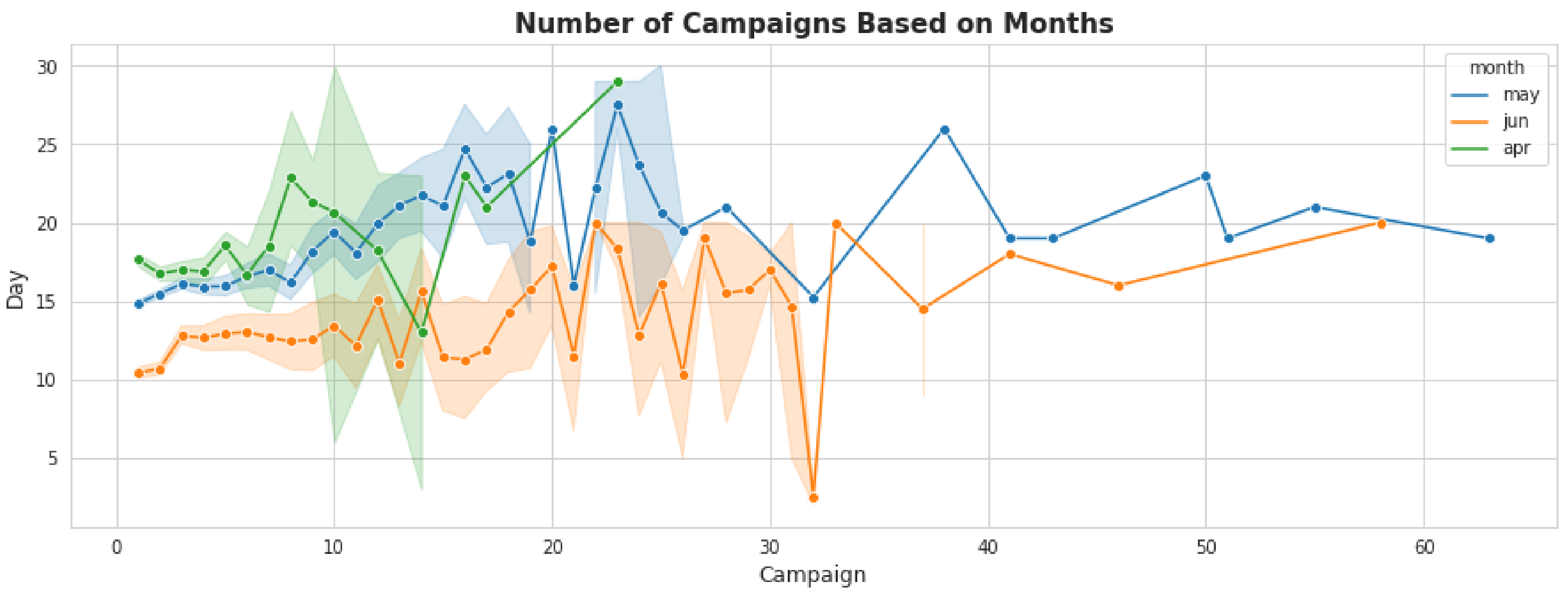
from pylab import rcParams %matplotlib inline rcParams['figure.figsize']=15,5

Q1: Create line plots to visualize the no. of campaigns for the given months, also perform the dot plots for the month April, May, June.

- figsize= H: 5inch, W: 15inch
- Title= FontS: 15
- X- & Y- axis = FontS: 12

Q1 done

```
plt.figure(figsize=(15,5))

'''apr= Bankdf[Bankdf['month']=='apr']
may= Bankdf[Bankdf['month']=='may']
jun= Bankdf[Bankdf['month']=='jun']
mon=pd.concat([may,apr,jun])'''

mon= Bankdf[Bankdf['month'].isin(['apr','may','jun'])]
sns.set_style('whitegrid')

#sns.relplot(data=mon, x='campaign', y='day',hue='month',kind='line',style='month',marker='o', dashes= False, legend='auto')

sns.lineplot(data=mon,x='campaign', y='day',hue='month',marker='o')
plt.title(label='Number of Campaigns Based on Months', fontdict={
    'fontsize':15,
    'fontweight':800
})
plt.xlabel(xlabel='Campaign',fontdict={
    'fontsize':12
})
plt.ylabel(ylabel='Day',fontdict={
    'fontsize':12
})
plt.legend(loc='upper right', title= 'month')
#plt.figlegend(mon['month'])
plt.show()
```



Histogram with Seaborn

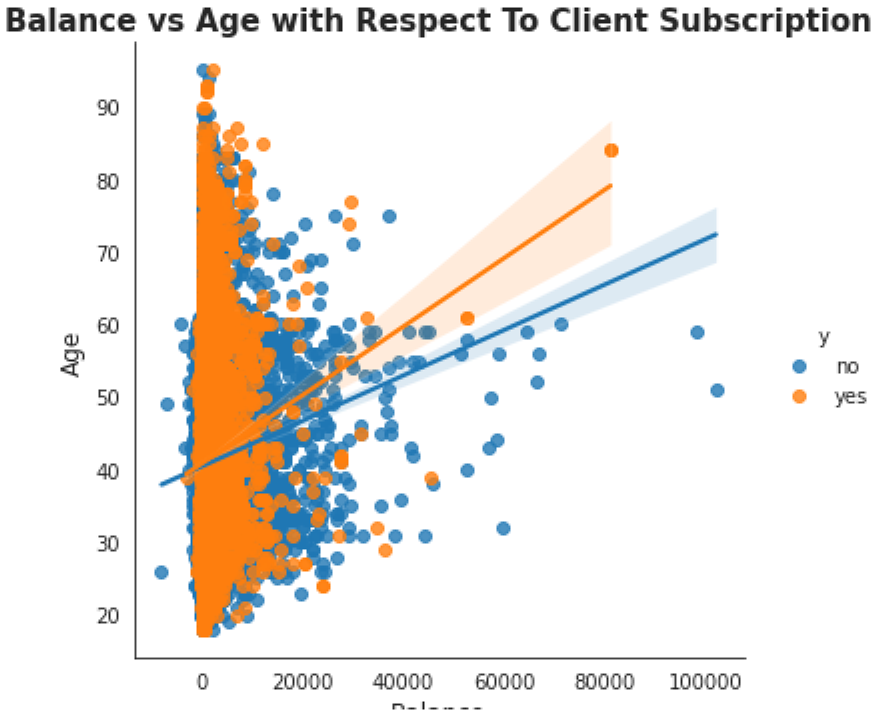
sns.distplot(Bankdf[['campaign','day']])

sns.regplot(data=Bankdf, x='balance', y='age') #Regression plot

Q5 done

```
sns.set_style('white')
plt.figure(figsize=(20,20))
sns.lmplot(data=Bankdf,x='balance',y='age', hue='y')
plt.title('Balance vs Age with Respect To Client Subscription', fontsize=15, fontweight=600)
plt.xlabel('Balance', fontsize=12)
plt.ylabel('Age', fontsize=12)
```

```
     Text(27.371197916666667, 0.5, 'Age')
     <Figure size 1440x1440 with 0 Axes>
```



```
sns.pairplot(Bankdf)
```

```
sns.boxplot(data=Bankdf, x='loan', y='age', palette='hls')
```

```
np.set_printoptions(precision=2)
```

```
sns.distplot(Bankdf['duration'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
<AxesSubplot:xlabel='duration', ylabel='Density'>
```

0.0035
0.0030
0.0025
0.0020
0.0015
Density

Q-6

- **But histplot is not supported in the mack test seaborn version**

duration

```
sns.set_style('white')
plt.figure(figsize=(10,5))
histcond=Bankdf[Bankdf['job'].isin(['blue-collar','retired','technician','admin','management'])]
sns.histplot(data=histcond,x='duration',hue='job',bins=80,
             hue_order=['blue-collar','retired','technician','admin','management'],
             multiple='layer',
            alpha=0.5,linewidth=0)
```

```
<AxesSubplot:xlabel='duration', ylabel='Count'>
```

Q2 Done

```
plt.figure(figsize=(10,10))
Corrdf=Bankdf[['age','balance','day','duration','campaign','pdays','previous']]
co=Corrdf.corr(method='pearson')
sns.heatmap(co,annot=True)
plt.title('Correlation Matrix on Bank Dataset', fontsize=15, fontweight=600)
```

```
Text(0.5, 1.0, 'Correlation Matrix on Bank Dataset')
```

```
print(co)
co['age']
co1=[]
for i in range(len(co.columns)):
    co1.append(round(co[co.columns[i]],i+1))
print(co1)
```

```
               age   balance       day  duration  campaign     pdays  previous
age       1.000000  0.097783 -0.009120 -0.004648  0.004760 -0.023758  0.001288
balance   0.097783  1.000000  0.004503  0.021560 -0.014578  0.003435  0.016674
day      -0.009120  0.004503  1.000000 -0.030206  0.162490 -0.093044 -0.051710
duration -0.004648  0.021560 -0.030206  1.000000 -0.084570 -0.001565  0.001203
campaign  0.004760 -0.014578  0.162490 -0.084570  1.000000 -0.088628 -0.032855
pdays    -0.023758  0.003435 -0.093044 -0.001565 -0.088628  1.000000  0.454820
previous  0.001288  0.016674 -0.051710  0.001203 -0.032855  0.454820  1.000000
[age         1.0
balance     0.1
day        -0.0
duration   -0.0
campaign    0.0
pdays      -0.0
previous    0.0
Name: age, dtype: float64, age         0.10
balance     1.00
day         0.00
duration    0.02
campaign   -0.01
pdays       0.00
previous    0.02
Name: balance, dtype: float64, age        -0.009
balance     0.005
day         1.000
duration   -0.030
campaign    0.162
pdays      -0.093
previous   -0.052
Name: day, dtype: float64, age       -0.0046
balance     0.0216
day        -0.0302
duration    1.0000
campaign   -0.0846
pdays      -0.0016
previous    0.0012
Name: duration, dtype: float64, age         0.00476
balance    -0.01458
day         0.16249
duration   -0.08457
campaign    1.00000
pdays      -0.08863
previous   -0.03286
Name: campaign, dtype: float64, age        -0.023758
balance     0.003435
day        -0.093044
duration   -0.001565
campaign   -0.088628
pdays       1.000000
previous    0.454820
Name: pdays, dtype: float64, age         0.001288
balance     0.016674
day        -0.051710
duration    0.001203
campaign   -0.032855
pdays       0.454820
previous    1.000000
Name: previous, dtype: float64]
```

sns.countplot(data=Bankdf, x='job', hue='housing') #part of catplot

df1=Bankdf df1.index=Bankdf['job'] df1.head()

df2=df1[['housing','loan']] df2

df2.plot(kind='bar')

df9= Bankdf[Bankdf.dtypes=='int'] df9.plot(kind='hist')

```
l_yes=Bankdf[Bankdf['loan']=='yes']['job'].value_counts()
l_no=len(Bankdf[Bankdf['loan']=='no'].value_counts())
h_yes=len(Bankdf[Bankdf['housing']=='yes'].value_counts())
h_no=len(Bankdf[Bankdf['housing']=='no'].value_counts())
print(l_yes)
```

```
     blue-collar      1684
     technician       1309
     management       1253
     admin.            991
     services          836
     entrepreneur      356
     retired           309
     self-employed     229
     housemaid         152
     unemployed        109
     student            12
     unknown             4
     Name: job, dtype: int64
```

```
Dfgroupjob= Bankdf.groupby(Bankdf['job'])
housing_count = Dfgroupjob['housing'].value_counts()
print(housing_count)
loan_count= Dfgroupjob['loan'].value_counts()
print(loan_count)
```

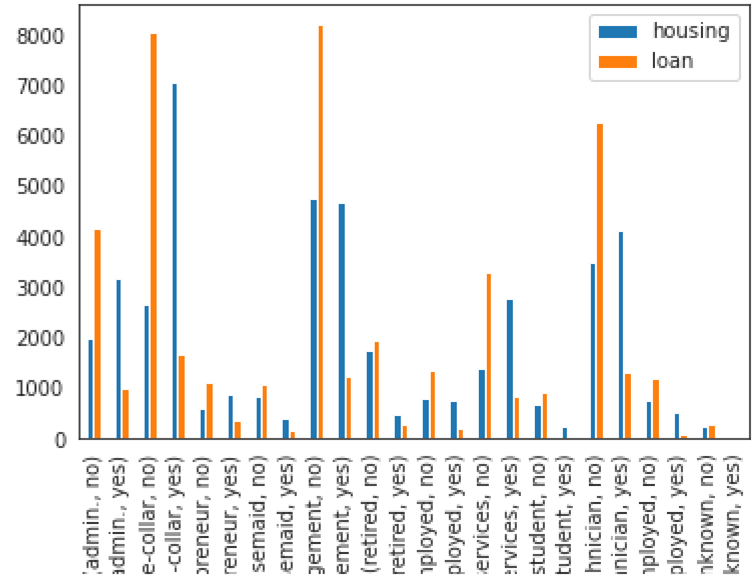
```
     job            housing
     admin.         yes        3182
                    no         1989
     blue-collar    yes        7048
                    no         2684
     entrepreneur   yes         869
                    no          618
     housemaid      no          842
                    yes         398
     management     no         4780
                    yes        4678
     retired        no         1773
                    yes         491
     self-employed  no          814
                    yes         765
     services       yes        2766
                    no         1388
     student        no          689
                    yes         249
     technician     yes        4115
                    no         3482
     unemployed     no          760
                    yes         543
     unknown        no          262
                    yes          26
     Name: housing, dtype: int64
     job            loan
     admin.         no      4180
                    yes      991
     blue-collar    no      8048
                    yes     1684
     entrepreneur   no      1131
                    yes      356
     housemaid      no      1088
                    yes      152
     management     no      8205
                    yes     1253
     retired        no      1955
                    yes      309
     self-employed  no      1350
                    yes      229
     services       no      3318
                    yes      836
     student        no       926
                    yes       12
     technician     no      6288
                    yes     1309
     unemployed     no      1194
                    yes      109
     unknown        no       284
                    yes        4
     Name: loan, dtype: int64
```

```
dataforbox= pd.DataFrame({
    'housing':housing_count,
    'loan':loan_count})
dataforbox
```

		housing	loan
job			
admin.	no	1989	4180
	yes	3182	991
blue-collar	no	2684	8048
	yes	7048	1684
entrepreneur	no	618	1131
	yes	869	356
housemaid	no	842	1088
	yes	398	152
management	no	4780	8205
	yes	4678	1253
retired	no	1773	1955
	yes	491	309
self-employed	no	814	1350
	yes	765	229
services	no	1388	3318
	yes	2766	836
student	no	689	926
	yes	249	12

```
'''index=Bankdf['job']
dataforbox= pd.DataFrame({
    'l_yes'
})'''
dataforbox.plot(kind='bar')
```

```
<AxesSubplot:xlabel='job,None'>
```



Q-3

```
'''hyesdf=Bankdf[Bankdf['housing']=='yes']
hnodf=Bankdf[Bankdf['housing']=='no']
lyesdf=Bankdf[Bankdf['loan']=='yes']
lnodf=Bankdf[Bankdf['loan']=='no']
hyesgrouped= hyesdf.groupby('job').count()
hyesgrouped.rename(columns = {'housing':'h_yes'}, inplace = True)
h_yes=hyesgrouped['h_yes']

hnogrouped= hnodf.groupby('job').count()
hnogrouped.rename(columns = {'housing':'h_no'}, inplace = True)
h_no=hnogrouped['h_no']

lyesgrouped= lyesdf.groupby('job').count()
lyesgrouped.rename(columns = {'loan':'l_yes'}, inplace = True)
l_yes=lyesgrouped['l_yes']

lnogrouped= lnodf.groupby('job').count()
lnogrouped.rename(columns = {'loan':'l_no'}, inplace = True)
l_no=lnogrouped['l_no']'''

l_yes=Bankdf[Bankdf['loan']=='yes']['job'].value_counts()
```

```
l_no=Bankdf[Bankdf['loan']=='no']['job'].value_counts()
h_yes=Bankdf[Bankdf['housing']=='yes']['job'].value_counts()
h_no=Bankdf[Bankdf['housing']=='no']['job'].value_counts()
print(h_no)

dataforbar= pd.DataFrame({
    'h_yes':h_yes,
    'h_no':h_no,
    'l_yes':l_yes,
    'l_no':l_no})
dataforbar.drop('unknown',axis=0, inplace=True)
dataforbar.sort_values(by='h_yes',ascending=False,inplace=True)
dataforbar.head()
```

```
management       4780
technician       3482
blue-collar      2684
admin.           1989
retired          1773
services         1388
housemaid         842
self-employed     814
unemployed        760
student           689
entrepreneur      618
unknown           262
Name: job, dtype: int64
```

	h_yes	h_no	l_yes	l_no
blue-collar	7048	2684	1684	8048
management	4678	4780	1253	8205
technician	4115	3482	1309	6288
admin.	3182	1989	991	4180
services	2766	1388	836	3318

Double-click (or enter) to edit

```
sns.set_style('white')
```

```
plt.figure(figsize=(20,15))
dataforbar.plot(kind='bar',figsize=(15,10),linewidth=0)
plt.legend(loc='upper right')  #bbox_to_anchor= (.815,1)
plt.title('Job and Loan')
plt.xlabel('Job Type')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
<Figure size 1440x1080 with 0 Axes>
```

Ques 4

```
plt.figure(figsize=(10,10))
dotdf= Bankdf[Bankdf['month'].isin(['jan','feb','mar','apr','may'])]
t=sns.scatterplot(data=dotdf, legend='auto', hue_order=['jan','feb','mar','apr','may'], hue='month' , x='campaign',y='duration',
                  palette=['green','gold','brown','DodgerBlue','Red'])
#plt.legend(loc='best')
plt.title('Campaign Vs Duration - Month Wise', fontsize=15, fontweight=600)
plt.xlabel('Campaign',fontsize=12)
plt.ylabel('Duration (in seconds)',fontsize=12)
plt.show()
```

```
Bankdf.info()
```

```
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 45211 entries, 0 to 45210
     Data columns (total 17 columns):
      #   Column     Non-Null Count  Dtype
     ---  ------     --------------  -----
      0   age        45211 non-null  int64
      1   job        45211 non-null  object
      2   marital    45211 non-null  object
      3   education  45211 non-null  object
      4   default    45211 non-null  object
      5   balance    45211 non-null  int64
      6   housing    45211 non-null  object
      7   loan       45211 non-null  object
      8   contact    45211 non-null  object
      9   day        45211 non-null  int64
      10  month      45211 non-null  object
      11  duration   45211 non-null  int64
      12  campaign   45211 non-null  int64
      13  pdays      45211 non-null  int64
      14  previous   45211 non-null  int64
      15  poutcome   45211 non-null  object
      16  y          45211 non-null  object
     dtypes: int64(7), object(10)
     memory usage: 5.9+ MB
```

```
s=(Bankdf.dtypes=='int64')
objcols=list(s[s].index)
matplothist=Bankdf[objcols]
matplothist.drop(['age'],axis=1, inplace=True)
matplothist.head()
```

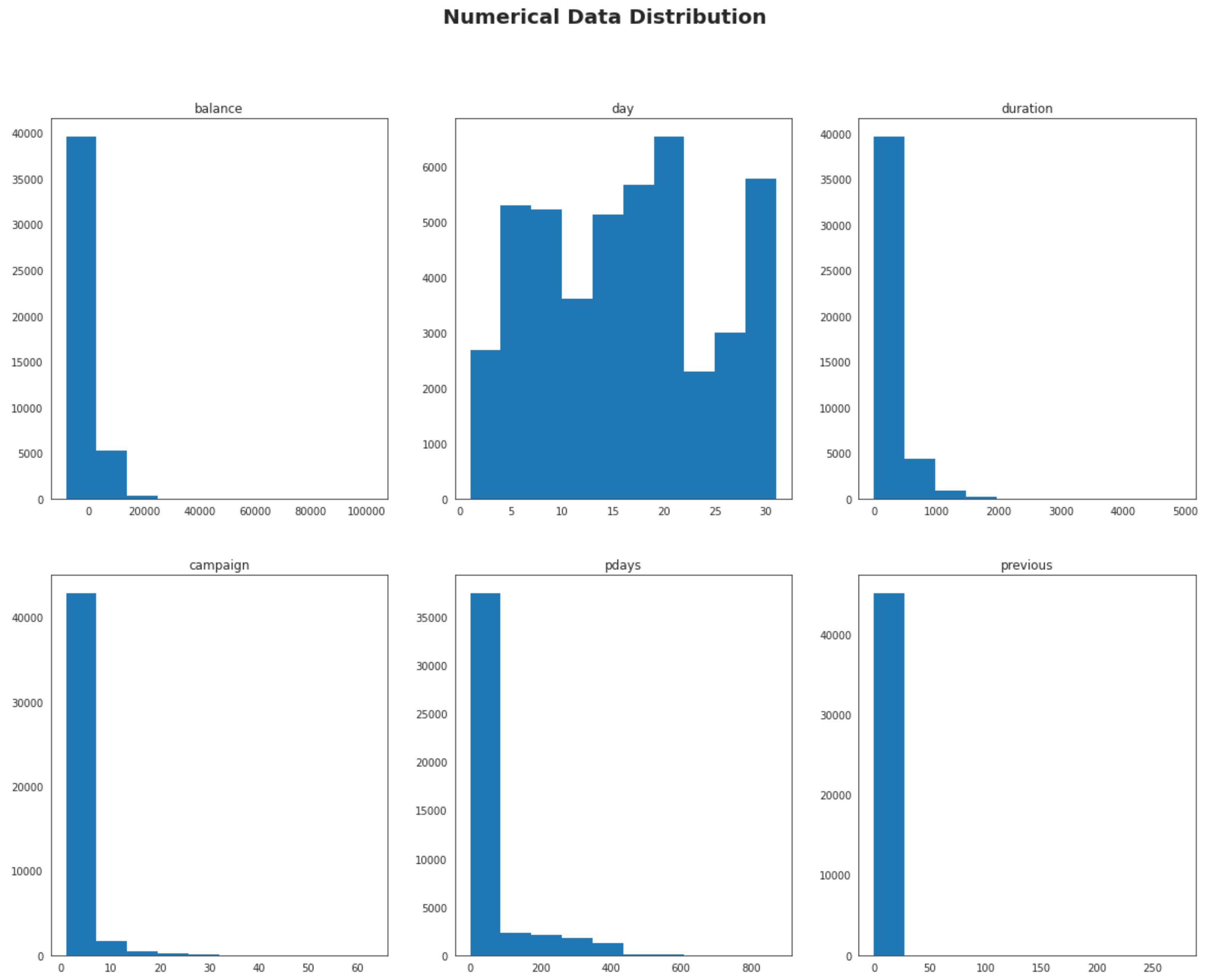
```
/opt/conda/lib/python3.7/site-packages/pandas/core/frame.py:4913: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
  errors=errors,
```

	balance	day	duration	campaign	pdays	previous
0	2143	5	261	1	-1	0
1	29	5	151	1	-1	0
2	2	5	76	1	-1	0
3	1506	5	92	1	-1	0

Q-7

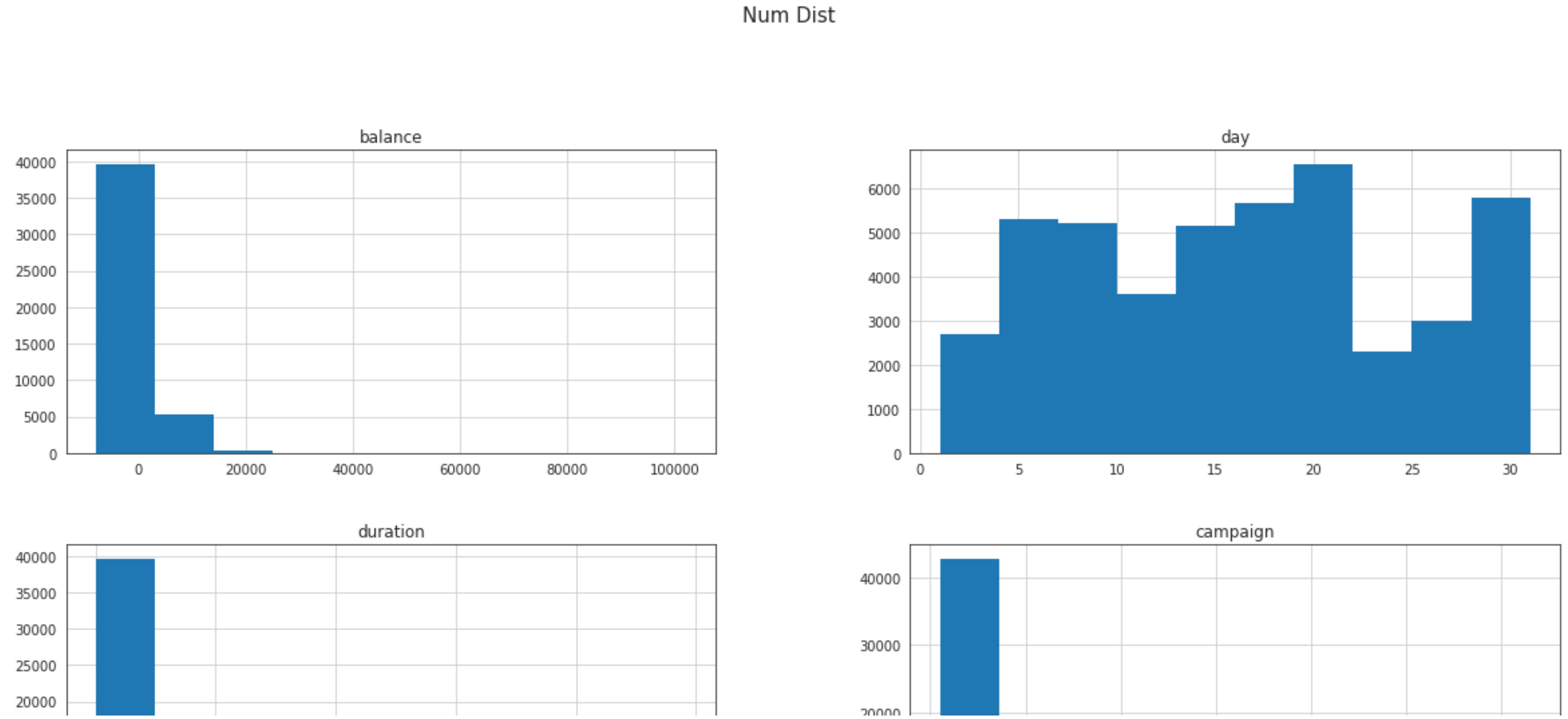
```
fig= plt.figure()
fig,((ax1,ax2,ax3),(ax4,ax5,ax6))=plt.subplots(2,3, figsize=(20,15))
fig.suptitle('Numerical Data Distribution',fontsize=20, fontweight=700)


ax1.hist(data=matplothist,x='balance',linewidth=0)
ax1.set_title('balance')
ax2.hist(data=matplothist,x='day',linewidth=0)
ax2.set_title('day')
ax3.hist(data=matplothist,x='duration',linewidth=0)
ax3.set_title('duration')
ax4.hist(data=matplothist,x='campaign',linewidth=0)
ax4.set_title('campaign')
ax5.hist(data=matplothist,x='pdays',linewidth=0)
ax5.set_title('pdays')
ax6.hist(data=matplothist,x='previous',linewidth=0)
ax6.set_title('previous')

'''fig.legend([l1, l2], labels=labels,
           loc="upper right")'''
#plt.subplots_adjust(right=0.9)
```

```
'fig.legend([l1, l2], labels=labels,\n           loc="upper right")'
<Figure size 432x288 with 0 Axes>
```



```
x=matplothist.hist(figsize=(20,15),linewidth=0)
plt.suptitle('Num Dist', fontsize=15)
```

```
Text(0.5, 0.98, 'Num Dist')
```



plt.subplots(2, 2, sharex=True, sharey=True) plt.figure(figsize=(10,10)) plt.subplot(6,1,1) plt.hist(data=matplothist,x='balance') plt.subplot(6,1,2) plt.hist(data=matplothist,x='day') plt.subplot(6,1,3) plt.hist(data=matplothist,x='duration') plt.subplot(6,2,1) plt.hist(data=matplothist,x='campaign') plt.subplot(6,2,1) plt.hist(data=matplothist,x='pdays') plt.subplot(6,2,3) plt.hist(data=matplothist,x='previous')

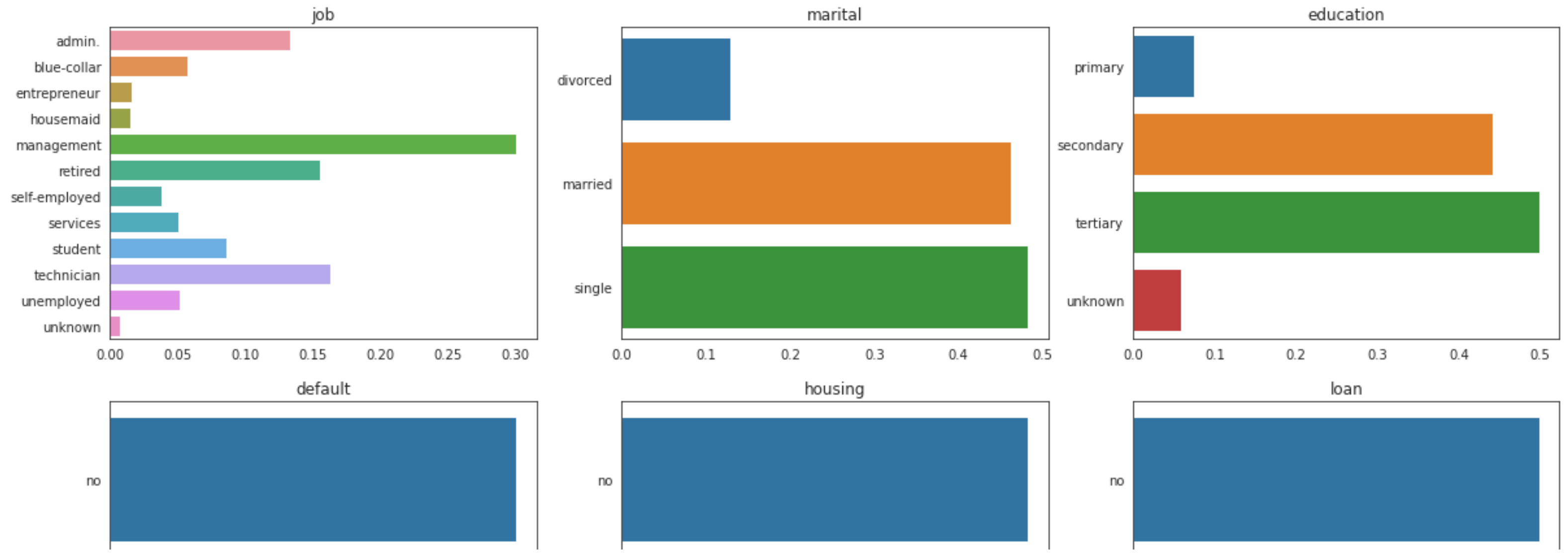
Q-8

- **Separate taget y**
- **Visulise the normalised relative frequency of the target class (y) per each category of each categorical column**
- **The difference of positive counts of each category on each column divided by total positive**
- **The difference of negative counts of each category on each column divided by total negative relative frequency.**
- **fig height 15, width 20**
- savefig(,bbox_inches='tight')
- **Using Seaborn Calculate**

s1= (Bankdf.dtypes=='object') reqcols=list(s1[s1].index) notreq=['default','housing','loan'] requiredcols= [i for i in reqcols if i not in notreq] requireddf= Bankdf[[r for r in Bankdf.columns if r in requiredcols]] requireddf.head()

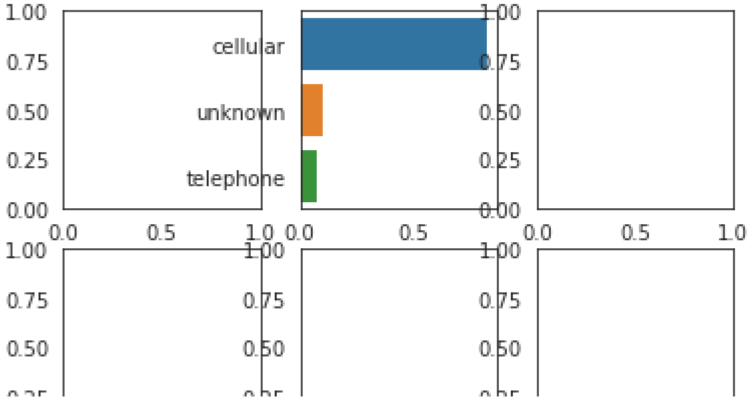
```
requireddf1=Bankdf[['job','marital','education','default','housing','loan','contact','month','poutcome','y']]
req1= requireddf1[requireddf1['poutcome']!='other']
requireddf=req1[req1['month']!='dec']
#fig= plt.figure()
fig,axes=plt.subplots(3,3, figsize=(20,15))

fig.suptitle('Normalised Relative Frequency',fontsize=20, fontweight=700)
y={}

for i in requireddf.columns:
    p_yes=requireddf[requireddf['y']=='yes'][i].value_counts()
    p_tye=len(requireddf[requireddf['y']=='yes'].value_counts())

    p_yest= (p_yes/p_tye)
#print(p_yest)

    p_no=(requireddf[requireddf['y']=='no'][i].value_counts())
    p_tne=len(requireddf[requireddf['y']=='no'])
    p_not= p_no/p_tne

    y[i]=(p_yest - p_not)

p_act=pd.DataFrame(y)
p_act.drop('y',axis=1, inplace=True)


for i in range(len(p_act.columns)):
    x=p_act[p_act.columns[i]].dropna(axis=0)
    df=x.to_frame()
    #print(df.head())
    if i<3:
        a=sns.barplot(data=df, x=df.columns[0], y=df.index,ax=axes[0,i%3])
        a.set(xlabel=None,title=df.columns[0])
    elif 3<=i<6:
        b=sns.barplot(data=df, x=df.columns[0], y=df.index,ax=axes[1,i%3])
        b.set(xlabel=None,title=df.columns[0])
    else:
        c=sns.barplot(data=df, x=df.columns[0], y=df.index,ax=axes[2,i%3])
        c.set(xlabel=None,title=df.columns[0])



#print(len(p_act.columns))

#sns.barplot(data=df, y='poutcome',x=df.index,linewidth=0, orient='h', ax=axes[0])
#sns.barplot(data=p_act, x='job',y=p_act.index,linewidth=0, ax=axes[0])
```

Normalised Relative Frequency



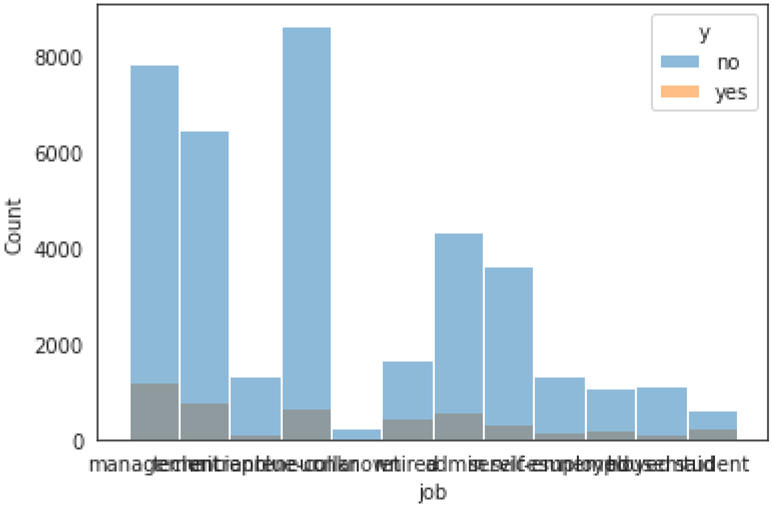
```
fig,axes=plt.subplots(2,3)

p_yes=Bankdf[Bankdf['y']=='yes']['contact'].value_counts()
p_tye=len(Bankdf[Bankdf['y']=='yes'].value_counts())

p_yest= (p_yes/p_tye)
#print(p_yest)

p_no=(Bankdf[Bankdf['y']=='no']['contact'].value_counts())
p_tne=len(Bankdf[Bankdf['y']=='no'])
p_not= p_no/p_tne

y= p_yest - p_not

p_act=pd.DataFrame({
    'yes':p_yest,
})

sns.barplot(data=p_act, x='yes',y=p_act.index, linewidth=0, ax=axes[0,1])
```

```
<AxesSubplot:xlabel='yes'>
```



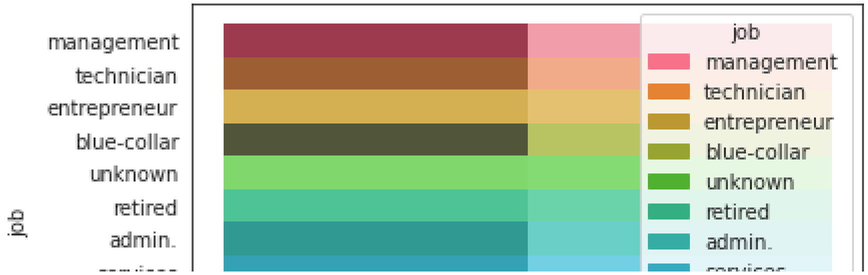
```
requim=requireddf[requireddf['month']!='dec']
requim[requim['month']=='nov'].head()

sns.histplot(data=requim, x='job',hue='y')
```

```
<AxesSubplot:xlabel='job', ylabel='Count'>
```



```
sns.histplot(data=Bankdf, x='y',y='job', hue='job')
```

```
<AxesSubplot:xlabel='y', ylabel='job'>
```



```
print(Bankdf['poutcome'].unique())
print(Bankdf['default'].unique())
pday=Bankdf['pdays'].unique()
pday.sort()
print(pday)
```

```
['unknown' 'failure' 'other' 'success']
['no' 'yes']
[ -1   1   2   3   4   5   6   7   8   9  10  12  13  14  15  17  18  19
  20  21  22  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  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74
  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92
  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110
 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146
 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164
 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182
 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218
 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236
 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254
 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272
 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290
 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308
 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326
 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344
 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362
 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398
 399 401 403 404 405 407 409 410 411 412 413 414 415 416 417 419 420 421
 422 424 425 426 427 428 430 431 432 433 434 435 436 437 439 440 442 444
 445 446 449 450 452 454 455 456 457 458 459 460 461 462 463 464 465 466
 467 469 470 472 474 475 476 477 478 479 480 481 484 485 486 489 490 491
 492 493 495 500 503 504 508 511 514 515 518 520 521 524 526 528 529 530
 531 532 535 536 541 542 543 544 547 550 551 553 555 557 558 561 562 578
 579 585 586 587 589 592 594 595 603 616 626 633 648 651 655 656 667 670
 674 680 683 686 687 690 701 717 728 745 749 756 760 761 769 771 772 774
 775 776 778 779 782 784 791 792 804 805 808 826 828 831 838 842 850 854
 871]
```