In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

1. Import data sets and Load data set

In [15]:
```
df=pd.read_csv('bank1.csv')  # to read the csv file
df.head(10)   # it shows the starting 10 rows
```

Out[15]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may
5	35	management	single	tertiary	no	747	no	no	cellular	23	feb
6	36	self-employed	married	tertiary	no	307	yes	no	cellular	14	may
7	39	technician	married	secondary	no	147	yes	no	cellular	6	may
8	41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may
9	43	services	married	primary	no	-88	yes	yes	cellular	17	apr

2. A. Identify the Features data types before entering into the analysis

In [3]:
```
df.dtypes   # helps to know the datatype of the columns
```

Out[3]:
```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

In [4]:
```
for i in df.columns:
    print(i)
    print(df[i].unique()) # this loop is performed to know the unique values in eve
    print("")
```

```
age
[30 33 35 59 36 39 41 43 20 31 40 56 37 25 38 42 44 26 55 67 53 68 32 49
 78 23 52 34 61 45 48 57 54 63 51 29 50 27 60 28 21 58 22 46 24 77 75 47
 70 65 64 62 66 19 81 83 80 71 72 69 79 73 86 74 76 87 84]

job
['unemployed' 'services' 'management' 'blue-collar' 'self-employed'
 'technician' 'entrepreneur' 'admin.' 'student' 'housemaid' 'retired'
 'unknown']

marital
['married' 'single' 'divorced']

education
['primary' 'secondary' 'tertiary' 'unknown']

default
['no' 'yes']

balance
[ 1787  4789  1350 ...  -333 -3313  1137]

housing
['no' 'yes']

loan
['no' 'yes']

contact
['cellular' 'unknown' 'telephone']

day
[19 11 16  3  5 23 14  6 17 20 13 30 29 27  7 18 12 21 26 22  2  4 15  8
 28  9  1 10 31 25 24]

month
['oct' 'may' 'apr' 'jun' 'feb' 'aug' 'jan' 'jul' 'nov' 'sep' 'mar' 'dec']

duration
[  79  220  185  199  226  141  341  151   57  313  273  113  328  261
   89  189  239  114  250  148   96  140  109  125  169  182  247  119
  149   74  897   81   40  958  354  150   97  132  765   16  609  106
  365  205   11  105   59  425  204  181 1018 1740   98  441  272  159
  295  314  579  554  323  227  134  223  155  130  630  164  268  380
  154  221   67  367   87  701  652   63  398  224  406   60  521  279
  203  201  372  391  165  231  291  233  473  736  337  553  345   65
    9  259  371  280  243  435  258    7  317   76  170  386   83   69
  564  588  779  281 1877   51   32  176  161  187   24   85  236   54
   71  489   39  455   86  190   45  168  194  103  333  102   92  213
  289   77  324   84   10   35   82  676   80  549  135  412  101  253
  166   18  147   14   61  377  152  382  543  240   48  471  285  301
  768 1337  403  139  196  115   17   95  198  654  256  834   20  178
  111  186  297  210  112  222  195  123  145  124  216  483  690  344
  673  144  246  361  375 1097  180  373  230   58   88  487   29  484
  262  644  699   49   64  121  197  331  138  312  120  526  211   62
  988  451 1030 1484  445  383  605  330  171  442  772  249  357  271
  783  472  395   56  641  429  157  162  799 1370   22  215 1017  298
  126    8  555  270  339  342 1434   30  397  620    6  209  419  283
  188  267  245 1065  207  456  131   94  567  153   53  234  108  208
  597  505  332  212  493  681  287  202   37   72  325 1212  319  514
  551  142  293  107  127 1816  200  418  387  156   47  265   31   28
  369  854   46  266  321   99  430  264  118  343    5  722  748  523
  421   15  502  193  347  468  388 1735  172  117  587  501  282  110
```

```
  104  378 1407  738   70  904  336  238  585   68 1713  218  661  566
  136  160   44  792   73   90  346  192  682  651  405  350   36  389
 3025  219  427  533   19  819  278  617   34  668   75  146  356  251
  352  184  568  260  447  426  174  284  428  237 1031  700  590   43
   27 1181  122  307  770  767  232  986   66  158  306  559  183  631
 1282 1199  244   55  290  385  133   91   25  275  632  100   41  446
  304  335  276   42  614  557 1663  510 1259  225  404 1015  761  464
  206  667  143  717   38  254  882  957  299  167  500  177  457  460
 1028  315  381  643  508  128  492  257  241  536  601 1168  277  364
  229  402  175  255  820  116  463  603  191 2087  754  303  288  891
  558  228  353  296  432 1130  305  274  860  420  756  968  408   13
  763  316   50    4   78  286  766  648  688   21  593  407  563   52
  803  396  637  945 1178  506  409  327  618  936  329  179  731  670
  318  415  137  349  263  671  452  163  586  650  610  747  252  883
  684  686 1060  724  424  712  753 1081  376  433  411 1083  757  524
  653   93  503  217  475  340  242  530   23  935  773  423  626  578
  248  528  785  952 1174  915  937  129 1063  758  574  847 1558  789
 1441  322 1504  537  611   26   12  235  796 1126  697  931 1034  362
  410  570  633  659  302  727  214  173  635  540 1210  486  646  326
  414  716  449  580  399 1029  755  619  606  971  348  594 1275  379
 1032  393  808  923  413  541  602  762  360  310  311  638  355  300
  417  308  657  434  488 1309 1056  908  401  827  735  691  461  669
 1473 1386  294  910  550 1366 1532  955  513 1236  809  482 1164  674
  709  436  374  309  363  422  358  640  439  476  480  517  993   33
  730  636  750  334  868  351 1689  607  485 1021  732  577  733  788
  863 1073  525  696  535  370  465  338  956  546  470  836  544  443
 1149  707 1451 1143  477  450  467  292  806  560  759 1183  598  466
  431  269  542  562  515 1165  547  780  916  474  509  679 1472  965
 1139  504  672  749  454  531  479  384  973  728  656  998  320  615
  612  664  394  877 1971 1258  599  655 1994  743  924  929  491  719
  565 1529  390 1467 1007  665  990  582  458  775  497  698  702  520
 1156  884 1521  359  527  994 1579  437 1225  865  569 1011  814  984
  392  595 1448  529 1006  622  494 1022 1044  781 1124  400  516 1516
  978  720  495  798  876  875  481  793 1117 1223 1101  744 2769  561
  715  639  538 1721 1608  725  519  490  907  680  977  959  693 2029
 1009  718  805  623  976  600  469 1010  634 1531  764  532  825  539
  816  821 1231  742 2456  721  777  548  830  645  723  746  869 1173
  624  518  815  857  921  627  800  366 1088  812  866 1151  873  592
  663  576 1476  951 1234 1263  660]

campaign
[ 1  4  2  5  3  6 18 10  9  7 12 14 13 24 11  8 29 32 16 22 15 30 25 21
 17 19 23 20 50 28 31 44]

pdays
[ -1 339 330 176 147 241 152 105 342 101   5  92  56 170 182 297 196 460
 137 367 145 169 207 266 288 168 345 436  90 183 146 335 347 119   7 271
 181  88 141 126  61 373 351 242  62  91 308 250 172 265  78  28  79   1
 188 167  89 164 462 209 321 254  94 364  96 356 149 363 275 325 341 260
 358  87 303  98 327 337 322 102  99 370  84 212  63  81 191 360 332  80
  85 247 150 175 382 261 336  58 206 112 199 133 208 253 135 278 140 298
 273 124 281 162 323 349 117   2 256 333 116 268 136 198 357 259 353 174
 371 205 246  69 315 110 461 184 270 127 187  64 130 346 100 352 808 113
 378 292 287 107 293 139 138 193 274  97 103 359 185 674 211 300 334 280
 479  95 262 362 225   3 366  60 190 368 122 343 131 365 299 115 316 180
 154 313 264 350  73 232 204 143 375 186 344 210 248 177 221 189 104 258
 305 171 120 317 178 386 118 404 374 282 179 284 227 291 173 871 238 294
 222 435 340 426 239  83 111 415 255 235 244  38 683 329  59 151 192 158
 338 388 165 348 197 295 109 484 326 369 397 414 319 474  93 249 272 355
 195  82 541 231 153 201 761 114 385 267 161 467  75 106 223 312 148 309
 283  86 166 160 450 500 311 123 159 687 224 361  74  76 286  77  57 219
 331 804 144 234]
```

```
previous
[ 0  4  1  3  2  5 20  7  6 10  9  8 18 19 12 13 11 14 15 24 17 22 23 25]

poutcome
['unknown' 'failure' 'other' 'success']

y
['no' 'yes']
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        4521 non-null   int64 
 1   job        4521 non-null   object
 2   marital    4521 non-null   object
 3   education  4521 non-null   object
 4   default    4521 non-null   object
 5   balance    4521 non-null   int64 
 6   housing    4521 non-null   object
 7   loan       4521 non-null   object
 8   contact    4521 non-null   object
 9   day        4521 non-null   int64 
 10  month      4521 non-null   object
 11  duration   4521 non-null   int64 
 12  campaign   4521 non-null   int64 
 13  pdays      4521 non-null   int64 
 14  previous   4521 non-null   int64 
 15  poutcome   4521 non-null   object
 16  y          4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
```

In [211…
```
df.describe(include='all')   # it describes the value present in all the column and
```

Out[211]:

	age	job	marital	education	default	balance	housing	loan	cont
count	4521.000000	4521	4521	4521	4521	4521.000000	4521	4521	4
unique	NaN	12	3	4	2	NaN	2	2	
top	NaN	management	married	secondary	no	NaN	yes	no	cell
freq	NaN	969	2797	2306	4445	NaN	2559	3830	2
mean	41.170095	NaN	NaN	NaN	NaN	1422.657819	NaN	NaN	N
std	10.576211	NaN	NaN	NaN	NaN	3009.638142	NaN	NaN	N
min	19.000000	NaN	NaN	NaN	NaN	-3313.000000	NaN	NaN	N
25%	33.000000	NaN	NaN	NaN	NaN	69.000000	NaN	NaN	N
50%	39.000000	NaN	NaN	NaN	NaN	444.000000	NaN	NaN	N
75%	49.000000	NaN	NaN	NaN	NaN	1480.000000	NaN	NaN	N
max	87.000000	NaN	NaN	NaN	NaN	71188.000000	NaN	NaN	N

2. B. Convert the datatypes which are wrongly identified according to the business(domain). Kindly use the User Defined function and loop to convert the data types once.

In [241…
```
df['marital'].astype('category')
```

Out[241]:
```
0       married
1       married
2        single
3       married
4       married
         ...
4516    married
4517    married
4518    married
4519    married
4520     single
Name: marital, Length: 4521, dtype: category
Categories (3, object): ['divorced', 'married', 'single']
```

2. C. Find and Remove missing if any. Use visualization to find the missing values or Use general method to find the missing values.

In [212…
```
df.isnull().sum()  # to find the null value we use is null but it will bool expres
```

Out[212]:
```
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
```

2. D. Find duplicates (if necessary)

In [213…
```
df.duplicated()   # it helps to find the duplicates in the tables
```

Out[213]:
```
0       False
1       False
2       False
3       False
4       False
        ...
4516    False
4517    False
4518    False
4519    False
4520    False
Length: 4521, dtype: bool
```

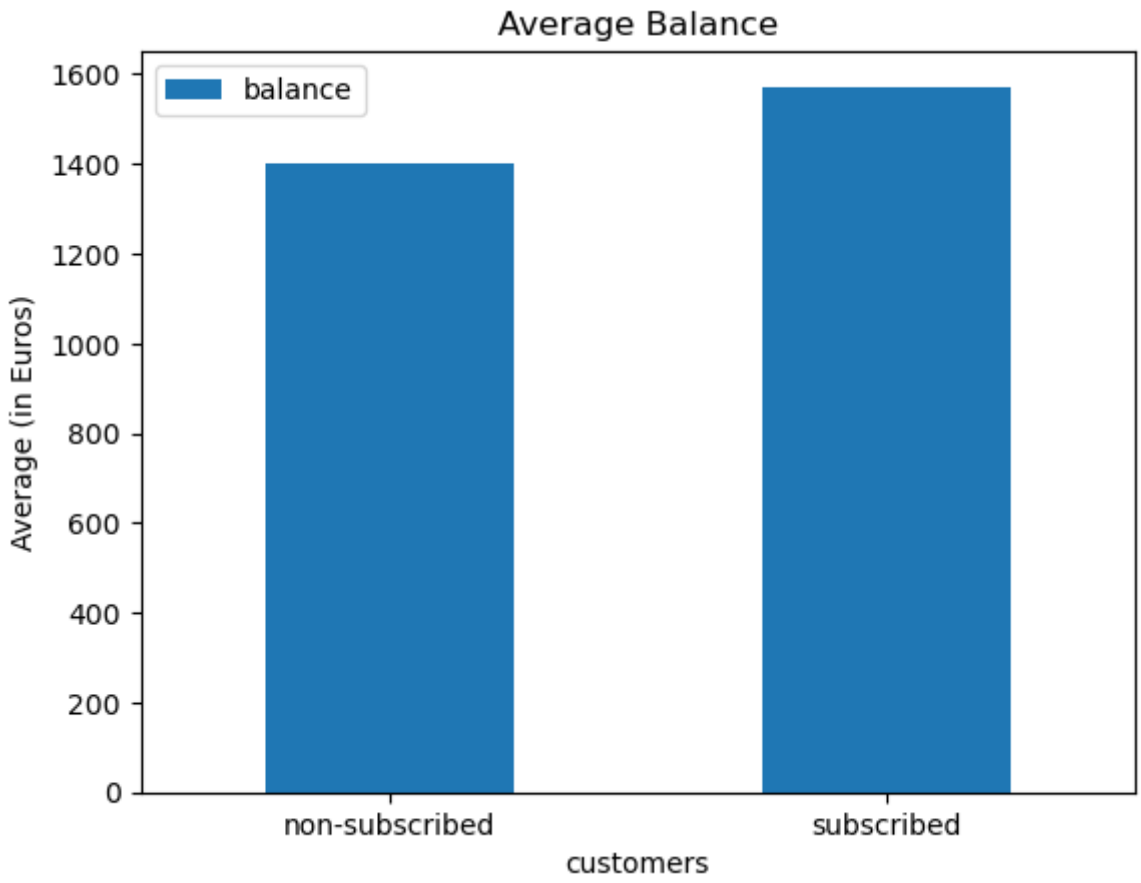
3. Find the average balance of the customer who belongs to the subscribed customer and non-subscribed customer and also use a related plot to show them in visualization.

In [14]:
```
df['y'].value_counts()
```

Out[14]:
```
non-subscribed    4000
subscribed         521
Name: y, dtype: int64
```

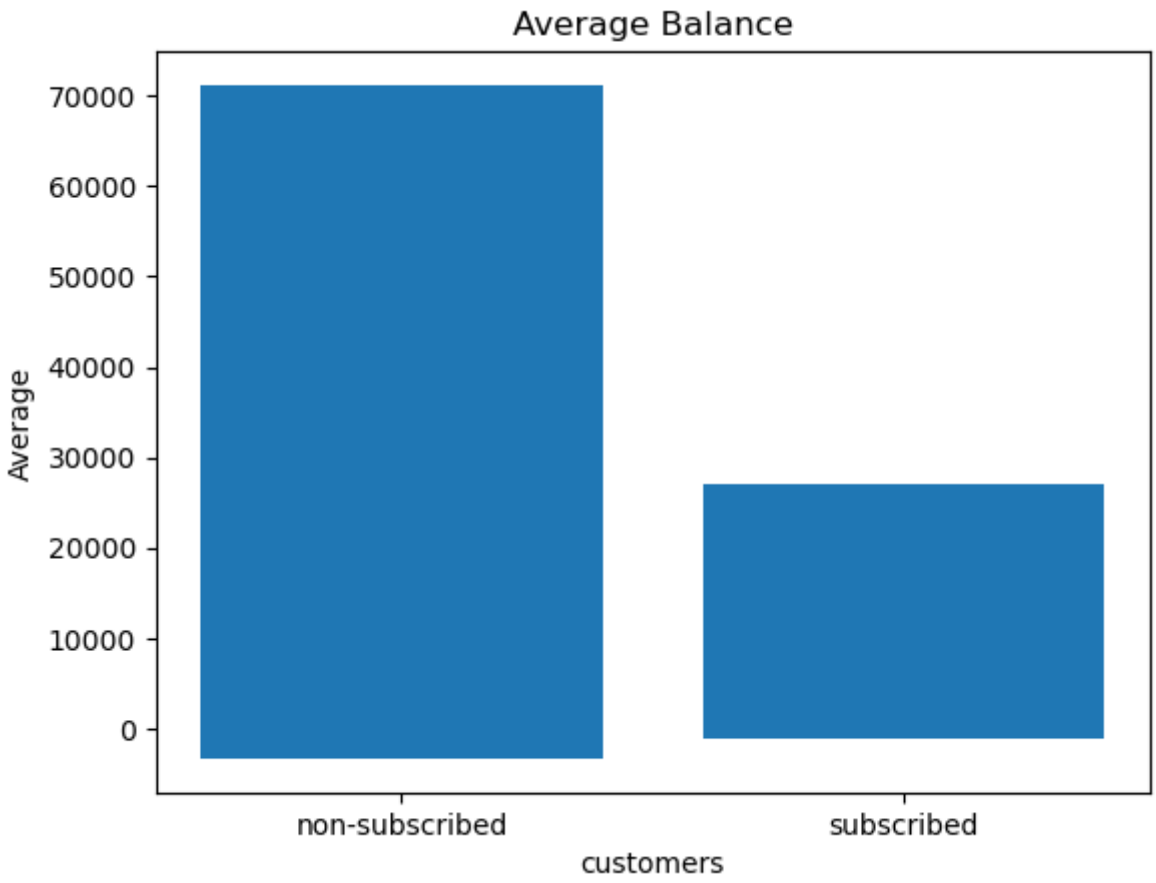
In [5]:
```
for i in df['y']:  # this loop is to change the value from "no" to non-subcribed ar
    if i=="no":
        df['y']=df["y"].replace(to_replace='no',value="non-subscribed")
    else:
        df['y']=df["y"].replace(to_replace='yes',value="subscribed")
```

In [6]:
```
pivot=pd.pivot_table(data=df,values='balance',index='y',aggfunc='mean')# to find th
pivot.plot(kind='bar')    #to plot the bar graph using one numerical data and one c
plt.title('Average Balance')
plt.xlabel('customers')
plt.ylabel('Average (in Euros)')
plt.xticks(rotation=0)
plt.show()
```



In [216…

```
plt.bar(df['y'],df['balance'])  # to plot the bar graph using one numerical data a
plt.title('Average Balance')
plt.xlabel('customers')
plt.ylabel('Average')
plt.show()
```

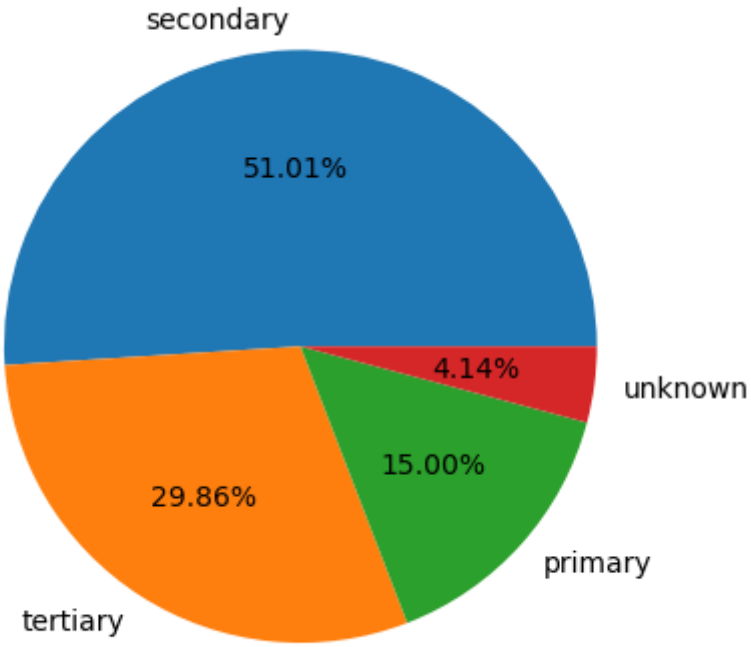


4.Use a pie plot to find the distribution(frequency) of the education. Make sure to add labels and show the percentage of each education distribution.

In [217…
```
ed_count=df['education'].value_counts() # to find the count of every values
ed_count
```

Out[217]:
```
secondary    2306
tertiary     1350
primary       678
unknown       187
Name: education, dtype: int64
```

In [218…
```
plt.pie(ed_count, labels= ed_count.index,autopct='%.2f%%')
plt.show()
```



5. Create a function that should be able to create a new feature(Variable) called season using the month column.

In [219…
```
def create_season_feature(df):
    df["season"]=df["month"].apply(lambda x: get_season(x))


def get_season(month):
    if month in ["dec","jan","feb"]:
        return "winter"
    elif month in ["march","apr","may"]:
        return "Spring"
    elif month in ["jun","jul","aug"]:
        return "Summer"
    else:
        return "Autumn"
```

In [220…
```
create_season_feature(df)
```

In [221…
```
df
```

Out[221]:

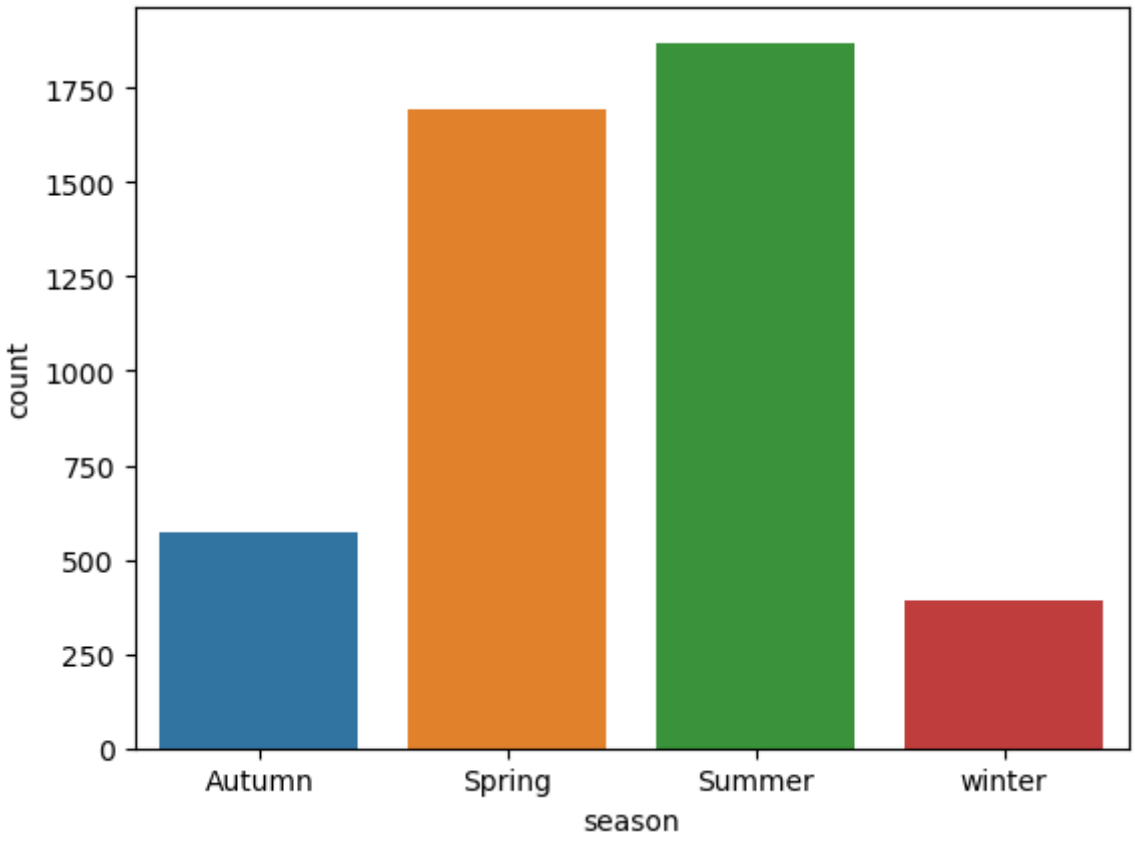
	age	job	marital	education	default	balance	housing	loan	contact	day	mo
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	r
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	r
...	...	...	...	...	...	...	...	...	...	...	
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9	r
4518	57	technician	married	secondary	no	295	no	no	cellular	19	a
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6	
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3	

4521 rows × 18 columns

6. Use the count plot with a variable that you created in the above question and also the Y variable to find the class distribution.

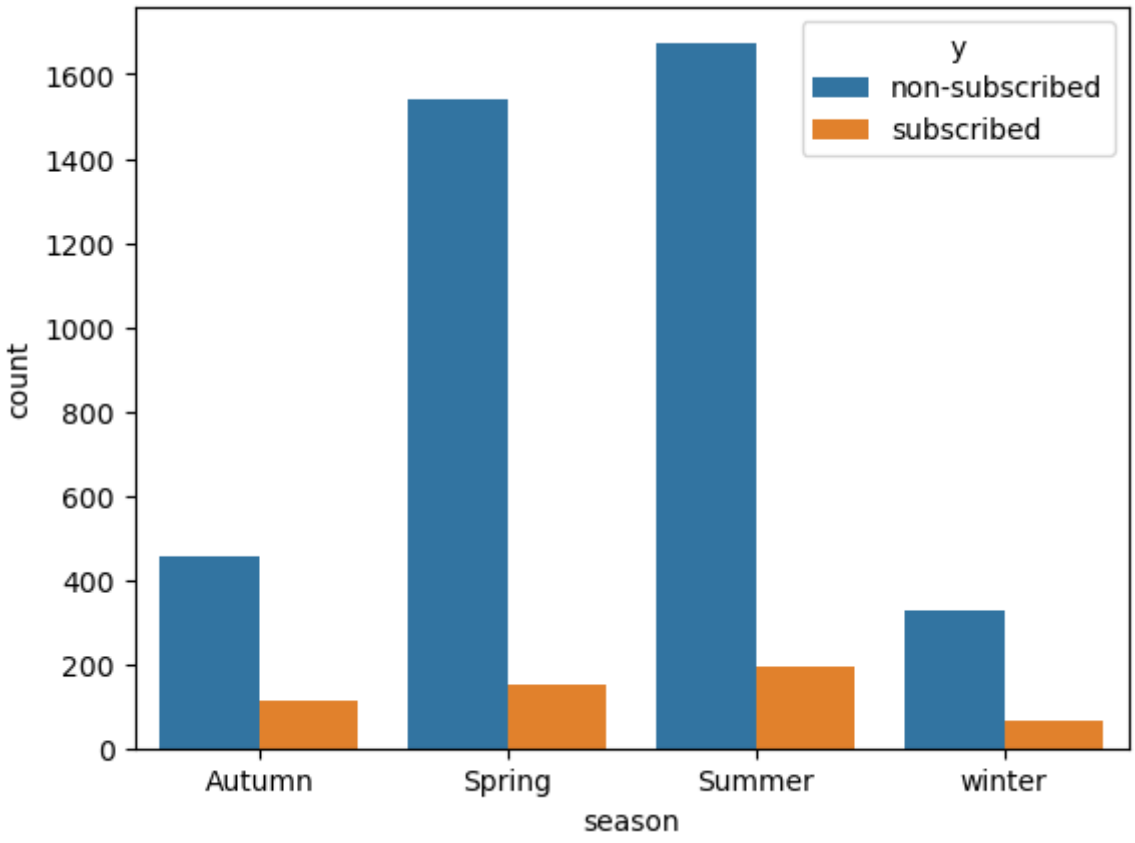
In [222…

```
sns.countplot(data=df,x='season')
plt.show()
```



In [223…

```
sns.countplot(data=df,x='season',hue='y')
plt.show()
```



7. Use the Pdays feature and find does it cause any effect on the subscription of the term using the bar

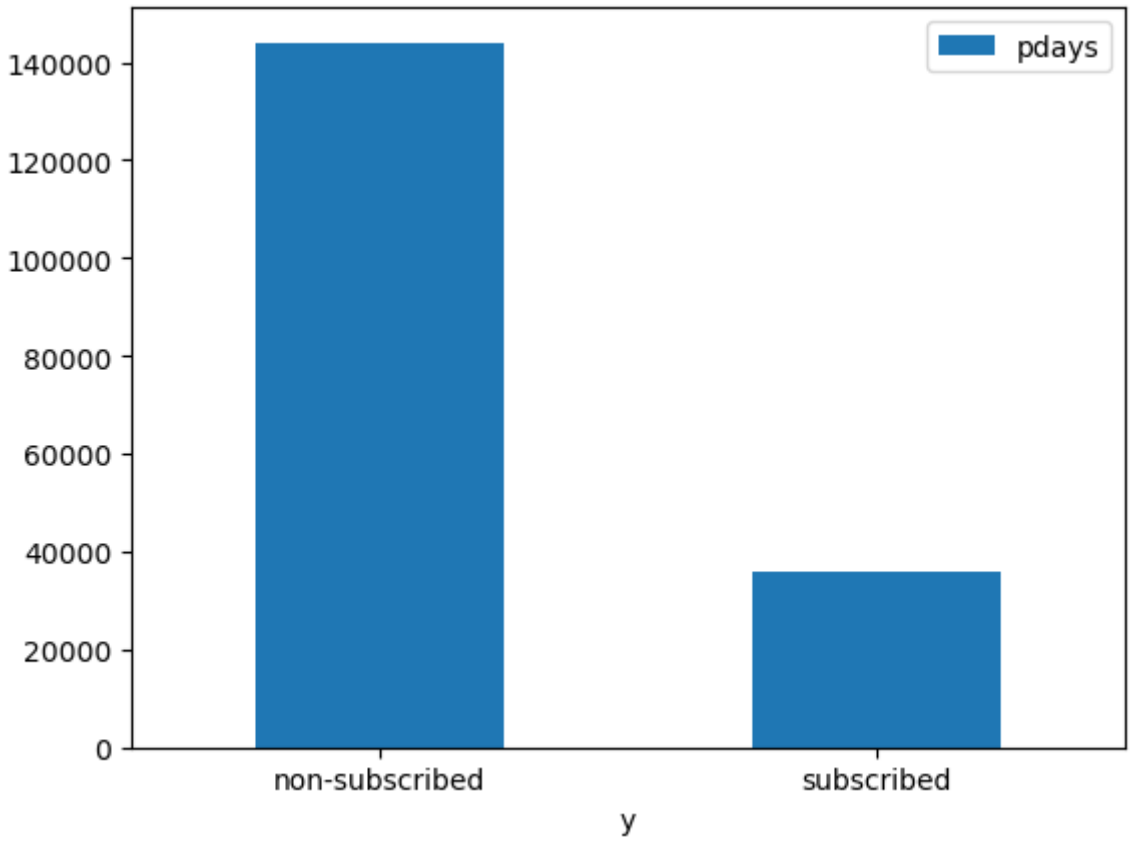
plot.

In [225…
```
pdf
```

Out[225]:

	pdays
y	
non-subscribed	144024
subscribed	35761

In [224…
```
pdf=pd.pivot_table(data=df,values='pdays',index='y',aggfunc='sum')
pdf.plot(kind='bar')
plt.xticks(rotation=0)
plt.show()
```



8. Replace the -1 as nan values for the P-days store.

In [226…
```
df['pdays'] = df['pdays'].replace(-1, np.nan)
```

9. Once you are done with question number 8, do the same analysis as question number 7. And observe the difference between question number 7 and question number 9.

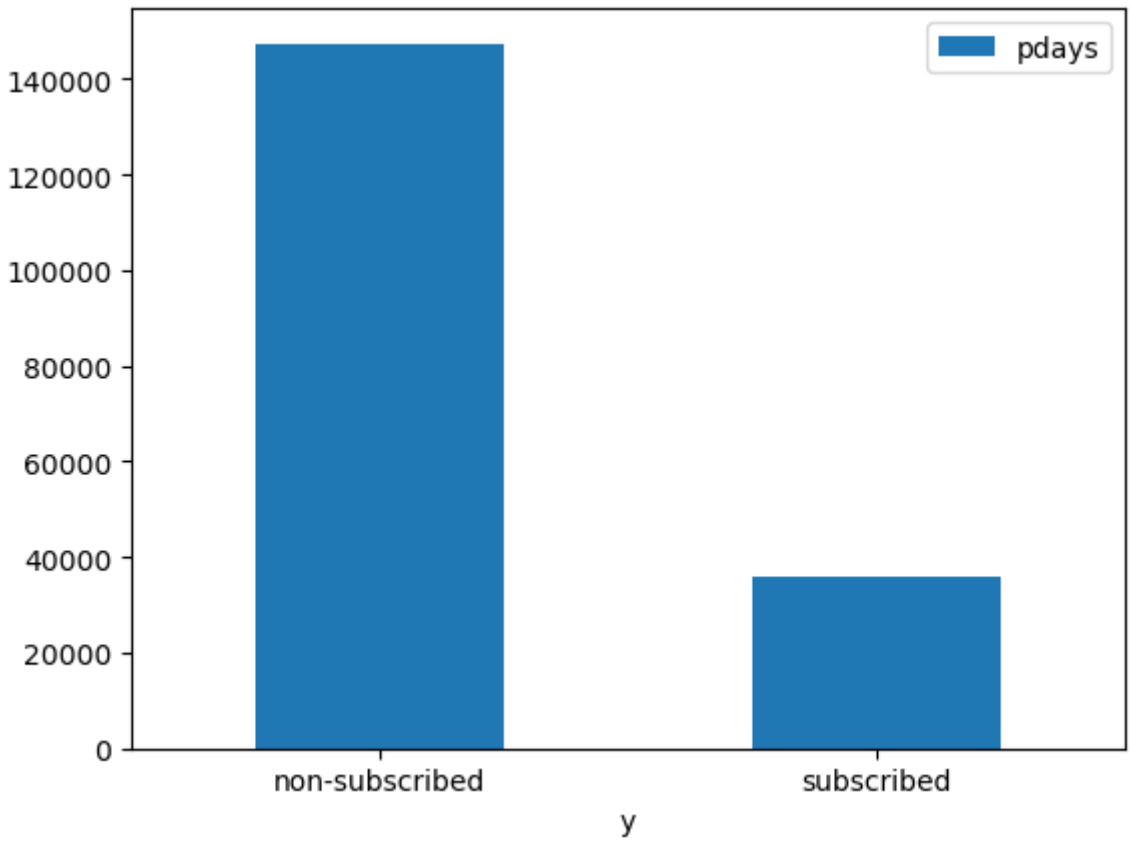
In [231…
```
pdf=pd.pivot_table(data=df,values='pdays',index='y',aggfunc='sum')
pdf
```

Out[231]:

y	pdays
non-subscribed	147392.0
subscribed	36098.0

In [230…

```
pdf.plot(kind='bar')
plt.xticks(rotation=0)
plt.show()
```



10. Does the customer take the term subscription who has less than 0 balance?

In [238…

```
df.head()
```

Out[238]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may

In [123…

```
neg = df[df['balance'] < 0]
neg['y'].value_counts()
```

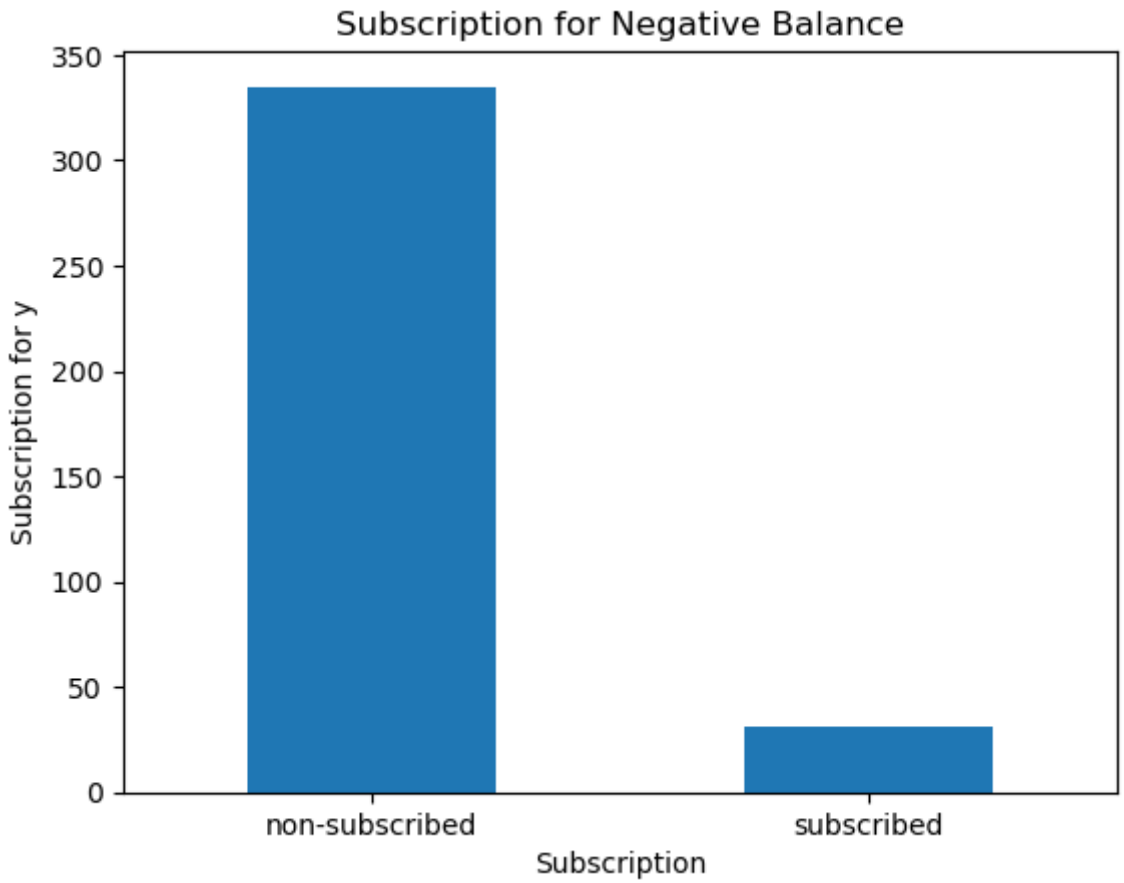
Out[123]:

```
non-subscribed    335
subscribed         31
Name: y, dtype: int64
```

In [233…

```
neg = df.loc[df['balance'] < 0]
subs = neg['y'].value_counts(normalize=False)

subs.plot(kind='bar')
plt.xlabel('Subscription')
plt.ylabel('Subscription for y')
plt.title('Subscription for Negative Balance')
plt.xticks(rotation=0)
plt.show()
```



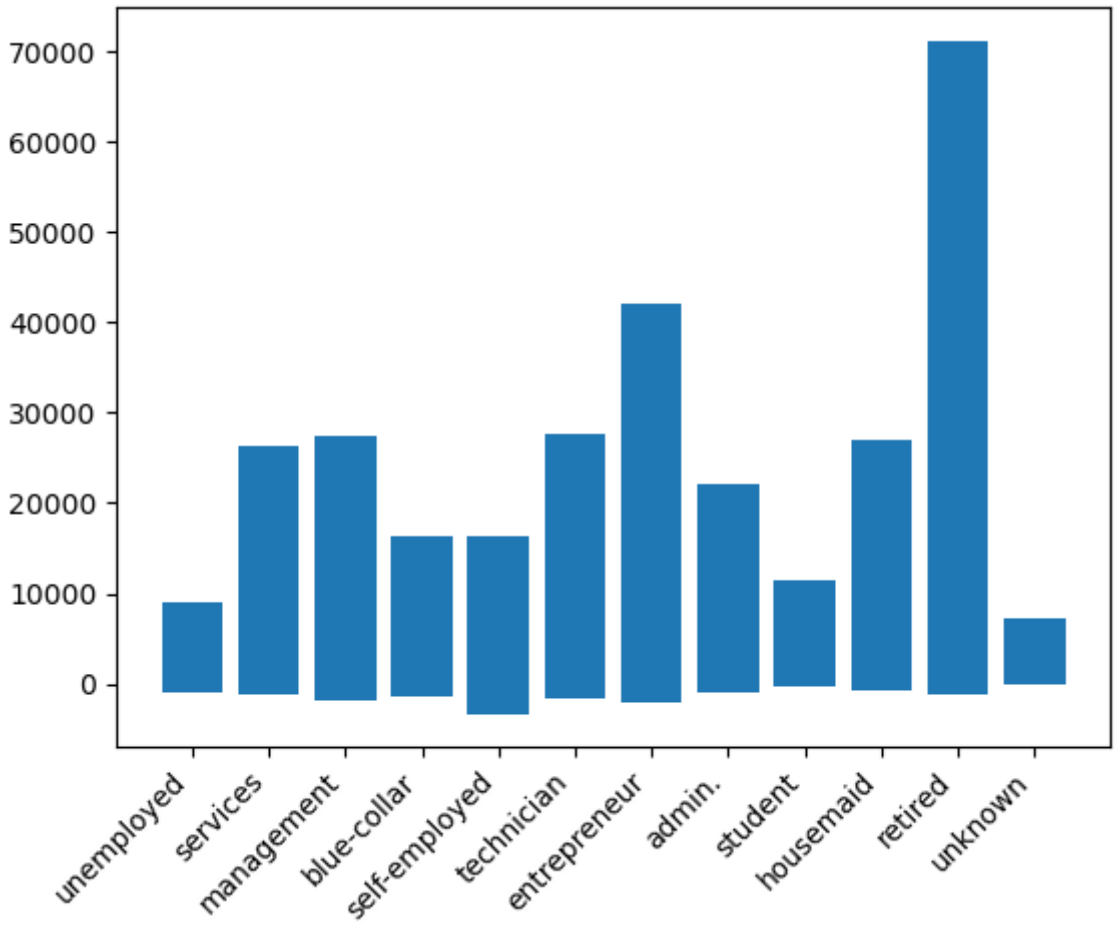
11. Use Pivot table to find the maximum balance for each type of job.

In [125…
```
pd.pivot_table(data = df, index="job",
               values = "balance",
               aggfunc = "max")
```

Out[125]:

job	balance
admin.	22171
blue-collar	16353
entrepreneur	42045
housemaid	26965
management	27359
retired	71188
self-employed	16430
services	26394
student	11555
technician	27733
unemployed	9019
unknown	7337

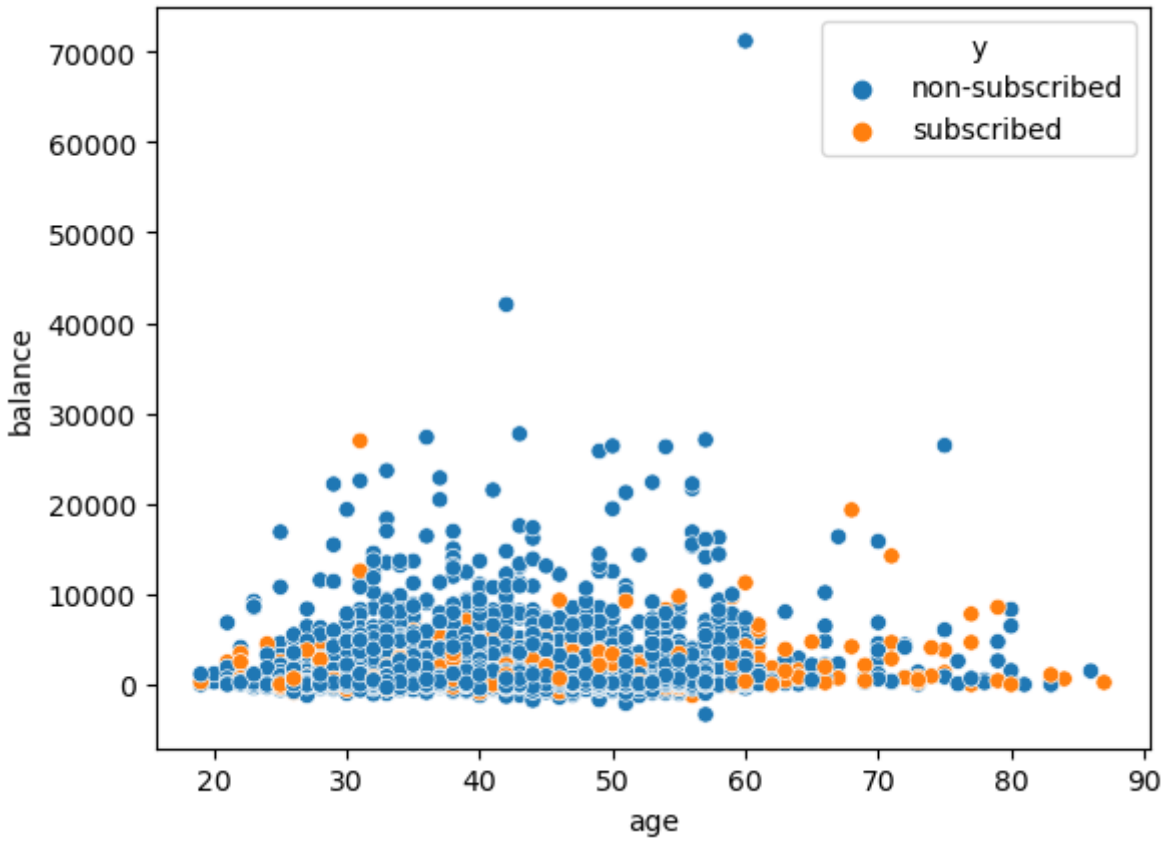
In [151…
```
plt.bar(data=df,x='job',height='balance')
plt.xticks(rotation=45, ha='right')
plt.show()
```



12. Use the Age, balance, and Y column to plot the scatter plot and find what kind of relationship Age and balance had, and See the points which belong 0 and 1 class and how they are distributed

In [126…
```
sns.scatterplot(data=df,x="age",y="balance",hue="y")
```

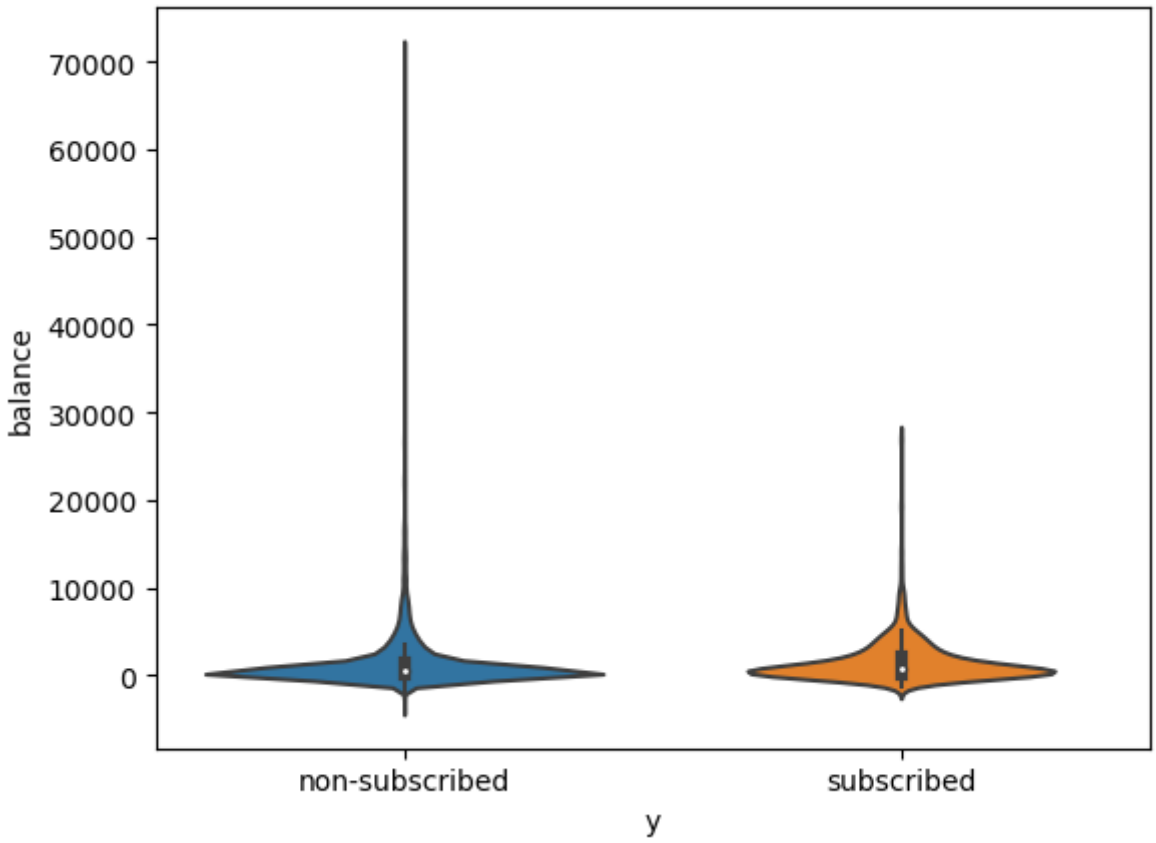
Out[126]:
```
<Axes: xlabel='age', ylabel='balance'>
```



13. Use the violin plot and also the box plot to find the distribution of the balance for each class of the Y column. And try to tell why we have a Violin plot and Box plot both rather than one.

In [234…

```
sns.violinplot(data = df,x = "y",
               y = "balance")
plt.show()
```



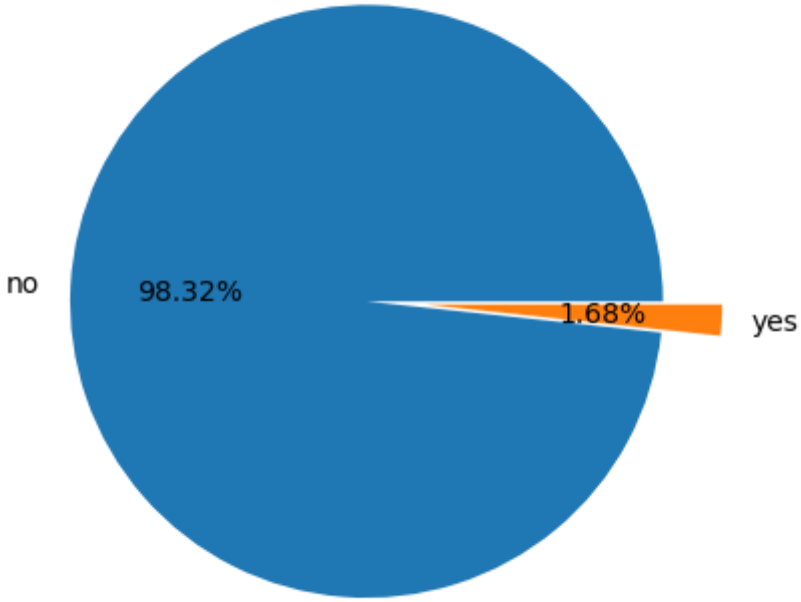
14. Use a pie plot to know the Proportion(distribution) of the defaulters and non-defaulters.

In [128…
```
defa=df["default"].value_counts()
```

In [129…
```
defa.values
```

Out[129]:
```
array([4445,   76], dtype=int64)
```

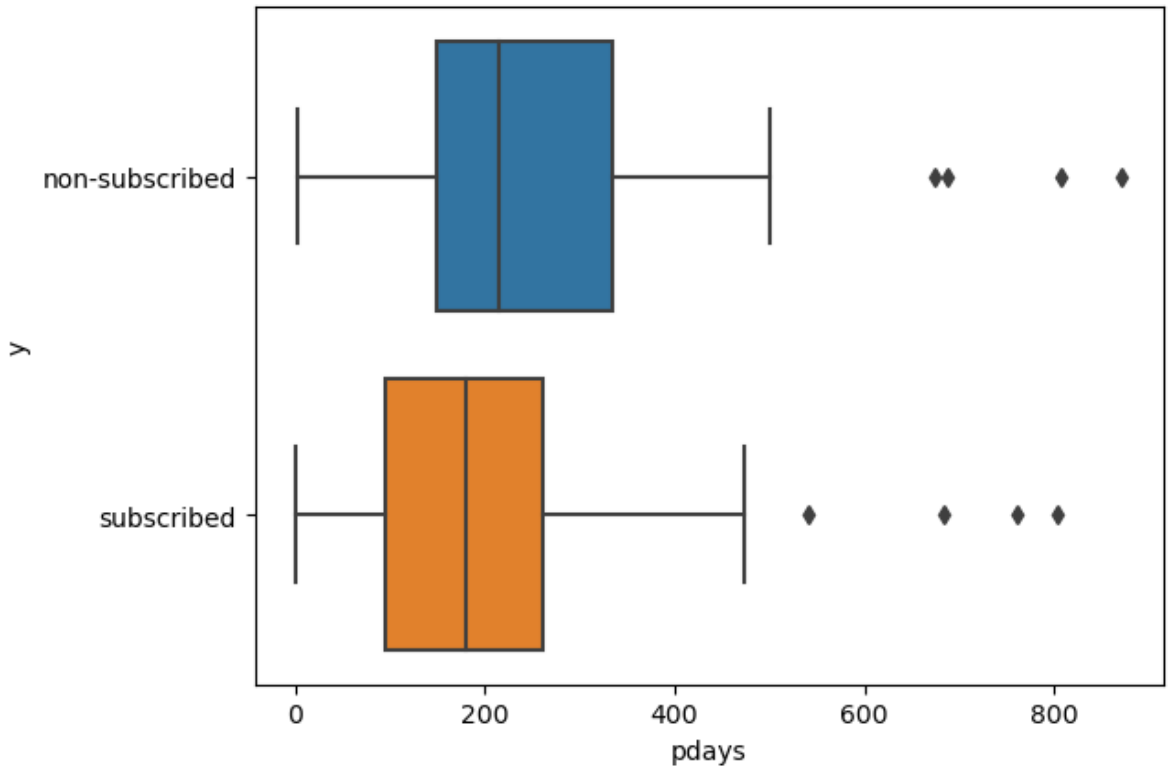
In [237…
```
plt.pie(defa.values,labels=defa.index,autopct="%.2f%%",explode=[0,0.2])
plt.show()
```



15. Use Box plot and strip plot to know the distribution of the Pdays with respect to Y classes and differentiate both plots.

In [138…

```
sns.boxplot(data = df,x = "pdays",
            y = "y")
plt.xticks(rotation=0)
plt.show()
```



In [255…

```
sns.stripplot(data = df,x = "y",
              y = "pdays")
plt.show()
```