EDA(Shreya Dwivedi)

- Hypothesis Test to verify: "People of age above 23 are less likely to subscrbe the term deposit."
- Hypothesis test to check if there is a relationship between p_days and previous.
- Hypothesis test to check if there is a correlation between our target variable and other categorical variables
- Univariate and Bivariate Analysis on Categorical Data

In [1]:

```
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import MinMaxScaler


import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:

```
# from google.colab import drive
# drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

In [3]:

```
# read in data
df_bank = pd.read_csv('D_G/bank.csv', delimiter=";")
df_bank_full = pd.read_csv('D_G/bank-full.csv', delimiter=";")
```

In [4]:

```
df_bank
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	
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	
...	...	...	...	...	...	...	...	...	...	...	
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	9	
4518	57	technician	married	secondary	no	295	no	no	cellular	19	
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 × 17 columns

In [5]:

```
df_bank_full
```

Out[5]:

	age	job	marital	education	default	balance	housing	loan	contact	day
0	58	management	married	tertiary	no	2143	yes	no	unknown	5
1	44	technician	single	secondary	no	29	yes	no	unknown	5
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5
4	33	unknown	single	unknown	no	1	no	no	unknown	5
...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17
45208	72	retired	married	secondary	no	5715	no	no	cellular	17
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17

45211 rows × 17 columns

In [6]:

```
df_bank_full.isnull().sum()
```

Out[6]:

```
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
```

In [7]:

```
df_bank.isnull().sum()
```

Out[7]:

```
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
```

In [8]:

```
#columns containing unknown df_bank
for column in df_bank:
  if 'unknown' in df_bank[column].values:
    print(column)
```

```
job
education
contact
poutcome
```

In [9]:

```
#columns containing unknown df_bank_full
for column in df_bank_full:
  if 'unknown' in df_bank_full[column].values:
    print(column)
```

```
job
education
contact
poutcome
```

In [10]:

```
# Predicting the unknown values using ML model
```

In [11]:

```
# Step1: Replacing unknown values with NaN values for df_bank
cols=['job','education','contact','poutcome']
mask = df_bank[cols].applymap(lambda x: x!="unknown")

df_bank[cols] = df_bank[cols].where(mask)
print (df_bank)
```

```
      age            job  marital  education default  balance housing loan  \
0      30     unemployed  married    primary      no     1787      no   no
1      33       services  married  secondary      no     4789     yes  yes
2      35     management   single   tertiary      no     1350     yes   no
3      30     management  married   tertiary      no     1476     yes  yes
4      59    blue-collar  married  secondary      no        0     yes   no
...   ...            ...      ...        ...     ...      ...     ...  ...
4516   33       services  married  secondary      no     -333     yes   no
4517   57  self-employed  married   tertiary     yes    -3313     yes  yes
4518   57     technician  married  secondary      no      295      no   no
4519   28    blue-collar  married  secondary      no     1137      no   no
4520   44   entrepreneur   single   tertiary      no     1136     yes  yes

       contact  day month  duration  campaign  pdays  previous poutcome   y
0     cellular   19   oct        79         1     -1         0      NaN  no
1     cellular   11   may       220         1    339         4  failure  no
2     cellular   16   apr       185         1    330         1  failure  no
3          NaN    3   jun       199         4     -1         0      NaN  no
4          NaN    5   may       226         1     -1         0      NaN  no
...        ...  ...   ...       ...       ...    ...       ...      ...  ..
4516  cellular   30   jul       329         5     -1         0      NaN  no
4517       NaN    9   may       153         1     -1         0      NaN  no
4518  cellular   19   aug       151        11     -1         0      NaN  no
4519  cellular    6   feb       129         4    211         3    other  no
4520  cellular    3   apr       345         2    249         7    other  no

[4521 rows x 17 columns]
```

In [12]:

```
#Replacing unknown values with NaN values for df_bank_full
cols=['job','education','contact','poutcome']
mask = df_bank_full[cols].applymap(lambda x: x!="unknown")

df_bank_full[cols] = df_bank_full[cols].where(mask)
print (df_bank_full)
```

```
        age            job   marital  education default  balance housing loan  \
0        58     management   married   tertiary      no     2143     yes   no
1        44     technician    single  secondary      no       29     yes   no
2        33   entrepreneur   married  secondary      no        2     yes  yes
3        47    blue-collar   married        NaN      no     1506     yes   no
4        33            NaN    single        NaN      no        1      no   no
...     ...            ...       ...        ...     ...      ...     ...  ...
45206    51     technician   married   tertiary      no      825      no   no
45207    71        retired  divorced    primary      no     1729      no   no
45208    72        retired   married  secondary      no     5715      no   no
45209    57    blue-collar   married  secondary      no      668      no   no
45210    37   entrepreneur   married  secondary      no     2971      no   no

         contact  day month  duration  campaign  pdays  previous poutcome    y
0            NaN    5   may       261         1     -1         0      NaN   no
1            NaN    5   may       151         1     -1         0      NaN   no
2            NaN    5   may        76         1     -1         0      NaN   no
3            NaN    5   may        92         1     -1         0      NaN   no
4            NaN    5   may       198         1     -1         0      NaN   no
...          ...  ...   ...       ...       ...    ...       ...      ...  ...
45206   cellular   17   nov       977         3     -1         0      NaN  yes
45207   cellular   17   nov       456         2     -1         0      NaN  yes
45208   cellular   17   nov      1127         5    184         3  success  yes
45209  telephone   17   nov       508         4     -1         0      NaN   no
45210   cellular   17   nov       361         2    188        11    other   no

[45211 rows x 17 columns]
```

In [13]:

```
sum=0
count=0
for i,val in enumerate(df_bank['pdays']):
  if val!=-1:
    sum+=val
    count+=(i+1)
mean=sum//count
for i,val in enumerate(df_bank['pdays']):
  if val==-1:
    df_bank['pdays'][i]=mean
df_bank
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: SettingWithCo
pyWarning:
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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

Out[13]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	
3	30	management	married	tertiary	no	1476	yes	yes	NaN	3	
4	59	blue-collar	married	secondary	no	0	yes	no	NaN	5	
...	...	...	...	...	...	...	...	...	...	...	
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	NaN	9	
4518	57	technician	married	secondary	no	295	no	no	cellular	19	
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 × 17 columns

In [14]:

```
sum=0
count=0
for i,val in enumerate(df_bank_full['pdays']):
  if val!=-1:
    sum+=val
    count+=(i+1)
mean=sum//count
for i,val in enumerate(df_bank_full['pdays']):
  if val==-1:
    df_bank_full['pdays'][i]=mean
df_bank_full
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: SettingWithCo
pyWarning:
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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

Out[14]:

	age	job	marital	education	default	balance	housing	loan	contact	day
0	58	management	married	tertiary	no	2143	yes	no	NaN	5
1	44	technician	single	secondary	no	29	yes	no	NaN	5
2	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5
3	47	blue-collar	married	NaN	no	1506	yes	no	NaN	5
4	33	NaN	single	NaN	no	1	no	no	NaN	5
...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17
45208	72	retired	married	secondary	no	5715	no	no	cellular	17
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17

45211 rows × 17 columns

In [15]:

```
df_bank_full.isnull().sum()
```

Out[15]:

```
age              0
job            288
marital          0
education     1857
default          0
balance          0
housing          0
loan             0
contact      13020
day              0
month            0
duration         0
campaign         0
pdays            0
previous         0
poutcome     36959
y                0
dtype: int64
```

In [16]:
```
df_bank.isnull().sum()
```

Out[16]:
```
age            0
job           38
marital        0
education    187
default        0
balance        0
housing        0
loan           0
contact     1324
day            0
month          0
duration       0
campaign       0
pdays          0
previous       0
poutcome    3705
y              0
dtype: int64
```

In [17]:
```
# Step 3: We will encode the categorical columns which do not have unknown va
independent_cat_col=['marital', 'default','housing','day', 'month','loan','y'
dependent_cat_col=['job','education']
```

In [18]:
```
# Dropping the columns contact and poutcome as contact wont contribute much t
df_bank=df_bank.drop(columns=['contact','poutcome'])
```

In [19]:
```
# Dropping the columns contact and poutcome as contact wont contribute much t
df_bank_full=df_bank_full.drop(columns=['contact','poutcome'])
```

In [20]:
```
# One-hot encoding df_bank
data_hot_encoded = pd.get_dummies(df_bank[independent_cat_col])

#Extract only the columns that didnt need to be encoded
data_other_cols = df_bank.drop(columns=independent_cat_col)

#Concatenate the two dataframes :
df_bank_out = pd.concat([data_hot_encoded, data_other_cols], axis=1)
```

In [21]:
```
df_bank_out
```

Out[21]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housing
0	19	0	1	0	1	0	
1	11	0	1	0	1	0	
2	16	0	0	1	1	0	
3	3	0	1	0	1	0	
4	5	0	1	0	1	0	
...	...	...	...	...	...	...	
4516	30	0	1	0	1	0	
4517	9	0	1	0	0	1	
4518	19	0	1	0	1	0	
4519	6	0	1	0	1	0	
4520	3	0	0	1	1	0	

4521 rows × 32 columns

In [22]:

```
df_bank['balance']
```

Out[22]:

```
0       1787
1       4789
2       1350
3       1476
4          0
        ... 
4516    -333
4517   -3313
4518     295
4519    1137
4520    1136
Name: balance, Length: 4521, dtype: int64
```

In [23]:

```
# One-hot encoding df_bank_full
data_hot_encoded = pd.get_dummies(df_bank_full[independent_cat_col])

#Extract only the columns that didnt need to be encoded
data_other_cols = df_bank_full.drop(columns=independent_cat_col)

#Concatenate the two dataframes :
df_bank_full_out = pd.concat([data_hot_encoded, data_other_cols], axis=1)
```

In [24]:
```
df_bank_full_out
```

Out[24]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housin
0	5	0	1	0	1	0	
1	5	0	0	1	1	0	
2	5	0	1	0	1	0	
3	5	0	1	0	1	0	
4	5	0	0	1	1	0	
...	...	...	...	...	...	...	
45206	17	0	1	0	1	0	
45207	17	1	0	0	1	0	
45208	17	0	1	0	1	0	
45209	17	0	1	0	1	0	
45210	17	0	1	0	1	0	

45211 rows × 32 columns

In [25]:
```
df_bank_out.isnull().sum()
```

Out[25]:
```
day                   0
marital_divorced      0
marital_married       0
marital_single        0
default_no            0
default_yes           0
housing_no            0
housing_yes           0
month_apr             0
month_aug             0
month_dec             0
month_feb             0
month_jan             0
month_jul             0
month_jun             0
month_mar             0
month_may             0
month_nov             0
month_oct             0
month_sep             0
loan_no               0
loan_yes              0
y_no                  0
y_yes                 0
age                   0
job                  38
education           187
balance               0
duration              0
campaign              0
pdays                 0
previous              0
dtype: int64
```

In [26]:
```
df_bank_full_out.isnull().sum()
```

Out[26]:

```
day                  0
marital_divorced     0
marital_married      0
marital_single       0
default_no           0
default_yes          0
housing_no           0
housing_yes          0
month_apr            0
month_aug            0
month_dec            0
month_feb            0
month_jan            0
month_jul            0
month_jun            0
month_mar            0
month_may            0
month_nov            0
month_oct            0
month_sep            0
loan_no              0
loan_yes             0
y_no                 0
y_yes                0
age                  0
job                288
education         1857
balance              0
duration             0
campaign             0
pdays                0
previous             0
dtype: int64
```

In [27]:

```
#  Make missing records as our Testing data.
#  Make non-missing records as our Training data.

# Testing data
```

In [28]:

```
# Applying Logistic Regression to predict the missing values
def LR(X_train, y_train, X_test):
  C_list=[10**0,10**1,10**2,10**3,1050,1075,1500, 1525,1575, 1590]
  list_score=[]
  max_C=float('-inf')
  X_train_norm = preprocessing.normalize(X_train)
  for i in C_list:
    model = LogisticRegression(penalty='l1', solver="saga", tol=0.1,C=i,multi_
    model = model.fit(X_train, y_train)
    score=cross_val_score(model,X_train_norm,y_train,cv=5,scoring='accuracy')
    mean_score=score.mean()
    list_score.append([i,mean_score])
  for i in list_score:
    if i[1]>max_C:
      max_C=i[1]
  model_l1_multi=LogisticRegression(penalty='l1', solver="saga", tol=0.1, C=m
  model_l1_multi.fit(X_train_norm,y_train)
  y_test=model_l1_multi.predict(X_test)
  return y_test
```

In [29]:

```
# Cannot use Naive Bayes as negative values are present
def Multinormial_Bayes(X_train, y_train, X_test):
  X_train_norm = preprocessing.normalize(X_train)
  mNB = MultinomialNB()
  model_NB = mNB.fit(X_train_norm, y_train)
  y_test=model_NB.predict(X_test)
  return y_test
```

In [30]:

```
def KnClassify(X_train, y_train, X_test):
  knn_clf=KNeighborsClassifier()
  knn_clf.fit(X_train,y_train)
  # y_test=[]
  y_test=knn_clf.predict(X_test)
  return y_test
```

In [31]:

```
def splittingTestTrainJob(df):
  df=df.drop(columns=['education'])
  df_train = df[~pd.isnull(df['job'])]
  df_test = df[pd.isnull(df['job'])]
  X_train_df_bank=df_train.drop(columns=['job'])
  y_train_df_bank=df_train['job']
  X_test_df_bank=df_test.drop(columns=['job'])
  return X_train_df_bank,y_train_df_bank,X_test_df_bank
```

In [32]:
```
def splittingTestTrainEducation(df_bank_out):
  df_bank_out_job=df_bank_out.drop(columns=['job'])
  df_bank_job_train = df_bank_out_job[~pd.isnull(df_bank_out_job['education']]
  df_bank_job_test = df_bank_out_job[pd.isnull(df_bank_out_job['education'])]
  X_train_df_bank=df_bank_job_train.drop(columns=['education'])
  y_train_df_bank=df_bank_job_train['education']
  X_test_df_bank=df_bank_job_test.drop(columns=['education'])
  return X_train_df_bank,y_train_df_bank,X_test_df_bank
```

In [33]:
```
def update_dataframe_job(X_test_df_bank,y_test_df_bank, X_train_df_bank,y_tra
  test_job=X_test_df_bank.copy()
  test_job=test_job.reset_index(drop=True)
  test_job['job']=y_test_df_bank
  test_job_train=X_train_df_bank.copy()
  test_job_train=test_job_train.reset_index(drop=True)
  y_train=y_train_df_bank.reset_index(drop=True)
  test_job_train["job"]=y_train
  new_df_bank_job=pd.concat([test_job_train,test_job])
  return new_df_bank_job
```

In [34]:
```
def update_dataframe_education(X_test_df_bank,y_test_df_bank, X_train_df_bank
  test_education=X_test_df_bank.copy()
  test_education=test_education.reset_index(drop=True)
  test_education['education']=y_test_df_bank
  test_education_train=X_train_df_bank.copy()
  test_education_train=test_education_train.reset_index(drop=True)
  y_train=y_train_df_bank.reset_index(drop=True)
  test_education_train["education"]=y_train
  new_df_bank_education=pd.concat([test_education_train,test_education])
  return new_df_bank_education
```

In [35]:
```
def new_data(new_df_bank_job,new_df_bank_education):
  new_df_bank=new_df_bank_job
  new_df_bank_education.reset_index()['education']
  new_df_bank['education']=new_df_bank_education.reset_index()['education']
  new_df_bank=new_df_bank.reset_index(drop=True)
  return new_df_bank
```

In [36]:
```
#Splitting data into test and train
X_train_df_bank,y_train_df_bank,X_test_df_bank=splittingTestTrainJob(df_bank_
```

In [37]:
```
# y_train_df_bank
```

In [38]:
```
# Predicting job using Logistic Multinomial Regression
y_test_df_bank=LR(X_train_df_bank, y_train_df_bank, X_test_df_bank)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but LogisticRegression was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

In [39]:
```
X_train_df_bank_ii,y_train_df_bank_ii,X_test_df_bank_ii=splittingTestTrainEdu
```

In [40]:
```
# Predicting education using Logistic Multinomial Regression
y_test_df_bank_ii=LR(X_train_df_bank_ii, y_train_df_bank_ii, X_test_df_bank_i
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but LogisticRegression was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

In [41]:
```
# Similarly for df_bak_full
X_train_df_bank_full,y_train_df_bank_full,X_test_df_bank_full=splittingTestTr
```

In [42]:
```
# len(X_train_df_bank_full)
```

In [43]:
```
# Predicting job using Logistic Multinomial Regression
y_test_df_bank_full=LR(X_train_df_bank_full, y_train_df_bank_full, X_test_df_
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but LogisticRegression was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

In [44]:
```
X_train_df_bank_full_ii,y_train_df_bank_full_ii,X_test_df_bank_full_ii=splitt
```

In [45]:
```
len(X_train_df_bank_full_ii)
```

Out[45]:
```
43354
```

In [46]:
```
# Predicting education using Logistic Multinomial Regression
y_test_df_bank_full_ii=LR(X_train_df_bank_full_ii, y_train_df_bank_full_ii, X
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but LogisticRegression was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

In [47]:
```
# Getting the dataset with predicted values for df_bank
new_df_bank_job=update_dataframe_job(X_test_df_bank, y_test_df_bank, X_train_
new_df_bank_job
```

Out[47]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housing_n
0	19	0	1	0	1	0	
1	11	0	1	0	1	0	
2	16	0	0	1	1	0	
3	3	0	1	0	1	0	
4	5	0	1	0	1	0	
...	...	...	...	...	...	...	.
33	27	0	1	0	1	0	
34	5	0	1	0	1	0	
35	5	0	0	1	1	0	
36	6	0	1	0	1	0	
37	11	0	1	0	1	0	

4521 rows × 31 columns

In [48]:
```
# test_education_train.isna().sum()
```

In [49]:
```
# new_df_bank_job.isna().sum()
```

In [50]:
```
new_df_bank_education=update_dataframe_education(X_test_df_bank_ii, y_test_df_
new_df_bank_education
```

Out[50]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housing_
0	19	0	1	0	1	0	
1	11	0	1	0	1	0	
2	16	0	0	1	1	0	
3	3	0	1	0	1	0	
4	5	0	1	0	1	0	
...	...	...	...	...	...	...	
182	16	0	1	0	1	0	
183	31	0	1	0	1	0	
184	2	0	1	0	1	0	
185	21	0	1	0	1	0	
186	16	0	1	0	1	0	

4521 rows × 31 columns

In [51]:

```
# new_df_bank_education.isna().sum()
```

In [52]:

```
# New Bank dataframe for df_bank
new_df_bank=new_data(new_df_bank_job,new_df_bank_education)
new_df_bank
```

Out[52]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housing
0	19	0	1	0	1	0	
1	11	0	1	0	1	0	
2	16	0	0	1	1	0	
3	3	0	1	0	1	0	
4	5	0	1	0	1	0	
...	...	...	...	...	...	...	
4516	27	0	1	0	1	0	
4517	5	0	1	0	1	0	
4518	5	0	0	1	1	0	
4519	6	0	1	0	1	0	
4520	11	0	1	0	1	0	

4521 rows × 32 columns

In [53]:

```
# new_df_bank.isna().sum()
```

In [54]:

```
# Similarly # Getting the dataset with predicted values for df_bank
```

In [55]:

```
len(y_train_df_bank_full)
```

Out[55]:

```
44923
```

In [56]:

```
new_df_bank_full_job=update_dataframe_job(X_test_df_bank_full, y_test_df_bank_
new_df_bank_full_job
```

Out[56]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housing_
0	5	0	1	0	1	0	
1	5	0	0	1	1	0	
2	5	0	1	0	1	0	
3	5	0	1	0	1	0	
4	5	0	1	0	1	0	
...	...	...	...	...	...	...	
283	7	0	1	0	1	0	
284	9	0	1	0	1	0	
285	11	0	0	1	1	0	
286	8	0	1	0	1	0	
287	16	0	1	0	1	0	

45211 rows × 31 columns

In [57]:

```
new_df_bank_full_education=update_dataframe_education(X_test_df_bank_full_ii,
new_df_bank_full_education
```

Out[57]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housing_
0	5	0	1	0	1	0	
1	5	0	0	1	1	0	
2	5	0	1	0	1	0	
3	5	0	1	0	1	0	
4	5	0	0	1	1	0	
...	...	...	...	...	...	...	
1852	27	0	1	0	1	0	
1853	8	0	1	0	1	0	
1854	8	0	1	0	1	0	
1855	9	0	0	1	1	0	
1856	16	0	1	0	1	0	

45211 rows × 31 columns

In [58]:

```
new_df_bank_full_education.isna().sum()
```

Out[58]:

```
day                  0
marital_divorced     0
marital_married      0
marital_single       0
default_no           0
default_yes          0
housing_no           0
housing_yes          0
month_apr            0
month_aug            0
month_dec            0
month_feb            0
month_jan            0
month_jul            0
month_jun            0
month_mar            0
month_may            0
month_nov            0
month_oct            0
month_sep            0
loan_no              0
loan_yes             0
y_no                 0
y_yes                0
age                  0
balance              0
duration             0
campaign             0
pdays                0
previous             0
education            0
dtype: int64
```

In [59]:

```
# New Bank dataframe for df_bank_full
new_df_bank_full=new_data(new_df_bank_full_job,new_df_bank_full_education)
new_df_bank_full
```

Out[59]:

	day	marital_divorced	marital_married	marital_single	default_no	default_yes	housin
0	5	0	1	0	1	0	
1	5	0	0	1	1	0	
2	5	0	1	0	1	0	
3	5	0	1	0	1	0	
4	5	0	1	0	1	0	
...	...	...	...	...	...	...	
45206	7	0	1	0	1	0	
45207	9	0	1	0	1	0	
45208	11	0	0	1	1	0	
45209	8	0	1	0	1	0	
45210	16	0	1	0	1	0	

45211 rows × 32 columns

In [60]:

```
new_df_bank_full.isna().sum()
```

Out[60]:
```
day                  0
marital_divorced     0
marital_married      0
marital_single       0
default_no           0
default_yes          0
housing_no           0
housing_yes          0
month_apr            0
month_aug            0
month_dec            0
month_feb            0
month_jan            0
month_jul            0
month_jun            0
month_mar            0
month_may            0
month_nov            0
month_oct            0
month_sep            0
loan_no              0
loan_yes             0
y_no                 0
y_yes                0
age                  0
balance              0
duration             0
campaign             0
pdays                0
previous             0
job                  0
education            0
dtype: int64
```

Using KNN Classifier

In [61]:
```
# le = preprocessing.LabelEncoder()
# y_train_df_bank=le.fit_transform(y_train_df_bank)
# y_train_df_bank=pd.DataFrame(y_train_df_bank)
```

In [62]:
```
# # Predicting job using KNN Classification
# y_test_df_bank_knn=KnClassify(X_train_df_bank, y_train_df_bank, X_test_df_b
# y_test_df_bank_knn
```

In [63]:
```
# y_test_df_bank_knn=le.inverse_transform(y_test_df_bank_knn)
# y_test_df_bank_knn
```

In [64]:
```
# y_train_df_bank_ii=le.fit_transform(y_train_df_bank_ii)
# y_train_df_bank_ii=pd.DataFrame(y_train_df_bank_ii)
```

In [65]:
```
# # Predicting education using KNN Classification
# y_test_df_bank_ii=KnClassify(X_train_df_bank_ii, y_train_df_bank_ii, X_test_
```

In [66]:
```
# # Similarly for df_bak_full
# y_train_df_bank_full=le.fit_transform(y_train_df_bank_full)
# y_train_df_bank_full=pd.DataFrame(y_train_df_bank_full)
# y_train_df_bank_full_ii=le.fit_transform(y_train_df_bank_full_ii)
# y_train_df_bank_full_ii=pd.DataFrame(y_train_df_bank_full_ii)
```

In [67]:
```
# Predicting job using KNN Classification
# y_test_df_bank_full_knn=KnClassify(X_train_df_bank_full, y_train_df_bank_fu
```

In [68]:
```
# # Predicting education using KNN Classification
# y_test_df_bank_full_ii=KnClassify(X_train_df_bank_full_ii, y_train_df_bank_
```

In [69]:
```
df_bank=df_bank.drop(['job','education'],axis=1).reset_index(drop=True)
df_bank['job']=new_df_bank['job']
df_bank['education']=new_df_bank['education']
df_bank
```

Out[69]:

	age	marital	default	balance	housing	loan	day	month	duration	campaign	pdays
0	30	married	no	1787	no	no	19	oct	79	1	0
1	33	married	no	4789	yes	yes	11	may	220	1	339
2	35	single	no	1350	yes	no	16	apr	185	1	330
3	30	married	no	1476	yes	yes	3	jun	199	4	0
4	59	married	no	0	yes	no	5	may	226	1	0
...	...	...	...	...	...	...	...	...	...	...	...
4516	33	married	no	-333	yes	no	30	jul	329	5	0
4517	57	married	yes	-3313	yes	yes	9	may	153	1	0
4518	57	married	no	295	no	no	19	aug	151	11	0
4519	28	married	no	1137	no	no	6	feb	129	4	211
4520	44	single	no	1136	yes	yes	3	apr	345	2	249

4521 rows × 15 columns

In [70]:

```
df_bank_full=df_bank_full.drop(['job','education'],axis=1).reset_index(drop=T
```

In [71]:

```
# for i in new_df_bank_full['job']:
#   print(i)
```

In [72]:

```
df_bank_full['job']=new_df_bank_full['job']
df_bank_full['education']=new_df_bank_full['education']
df_bank_full
```

Out[72]:

	age	marital	default	balance	housing	loan	day	month	duration	campaign	pday
0	58	married	no	2143	yes	no	5	may	261	1	(
1	44	single	no	29	yes	no	5	may	151	1	(
2	33	married	no	2	yes	yes	5	may	76	1	(
3	47	married	no	1506	yes	no	5	may	92	1	(
4	33	single	no	1	no	no	5	may	198	1	(
...	...	...	...	...	...	...	...	...	...	...	..
45206	51	married	no	825	no	no	17	nov	977	3	(
45207	71	divorced	no	1729	no	no	17	nov	456	2	(
45208	72	married	no	5715	no	no	17	nov	1127	5	18
45209	57	married	no	668	no	no	17	nov	508	4	(
45210	37	married	no	2971	no	no	17	nov	361	2	18

45211 rows × 15 columns

In [73]:

```
# df_bank.isna().sum()
```

In [74]:

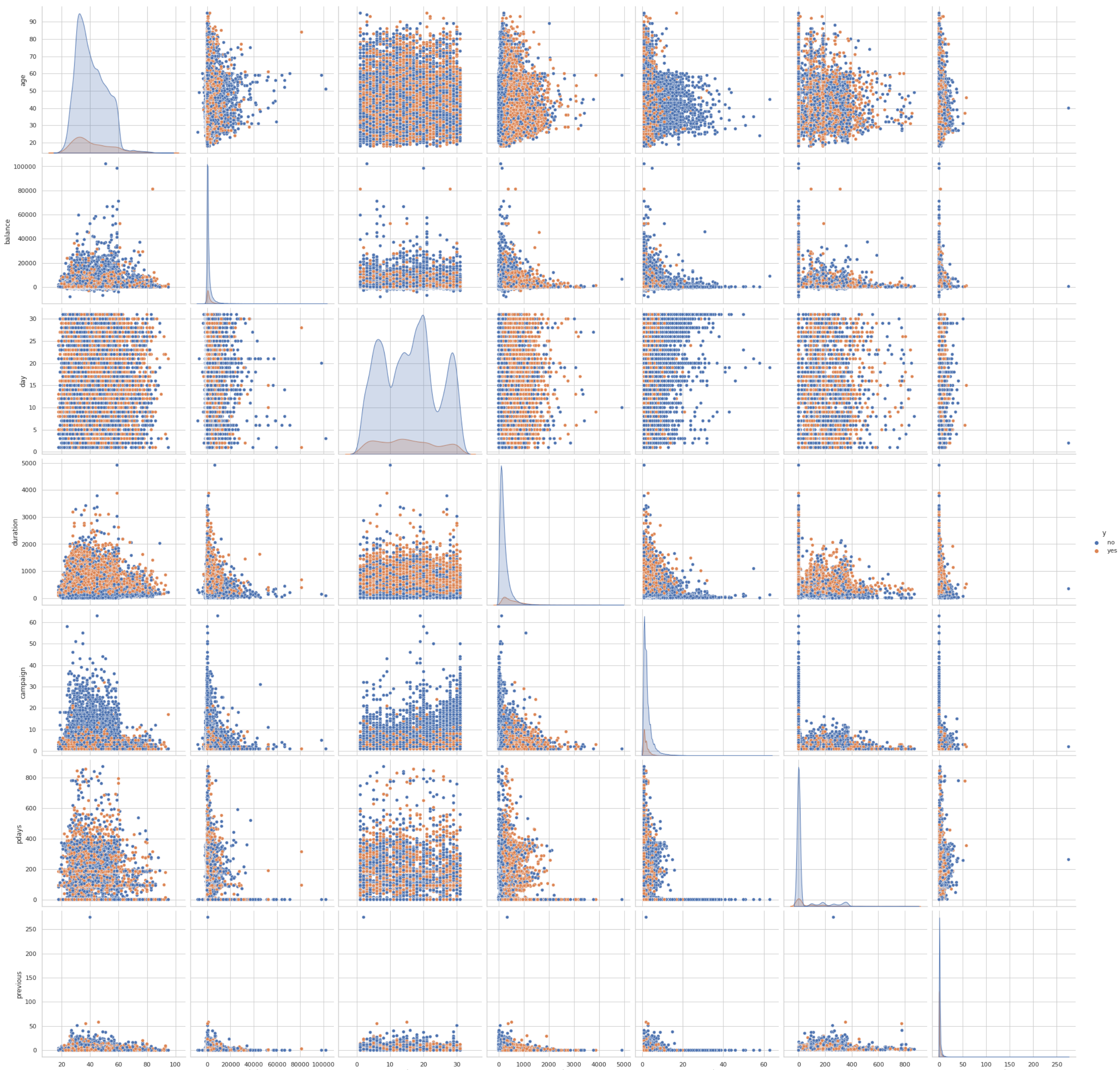
```
# df_bank_full.isna().sum()
```

Out[74]:

```
age          0
marital      0
default      0
balance      0
housing      0
loan         0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
y            0
job          0
education    0
dtype: int64
```

In [75]:
```
def draw_paiplot(df):
#Paiplot for among all the independent features
    sns.set(rc={'figure.figsize':(100,100)})
    sns.set_style("whitegrid");
    sns.pairplot(df, hue="y", height=4);
    plt.show()
```

In [76]:
```
draw_paiplot(df_bank_full)
```



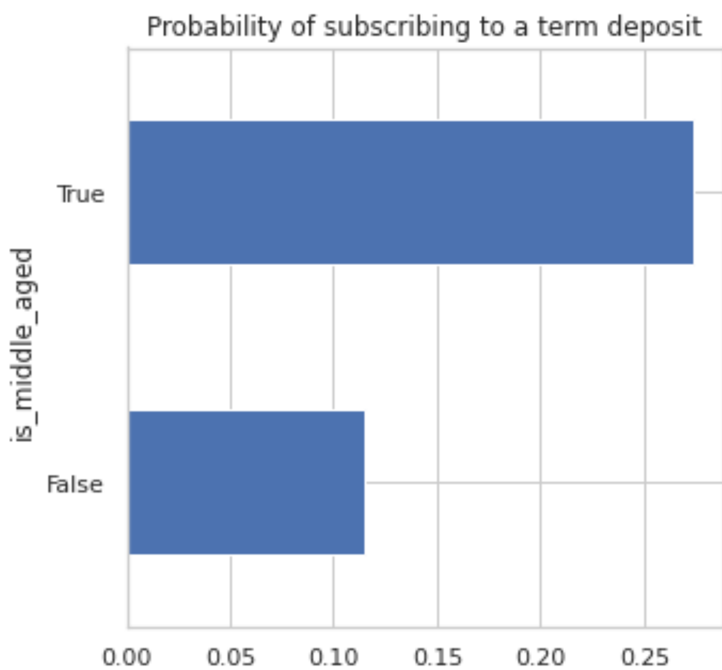
From the above graphs its difficult to draw much insights about the correlation between the features. So for that purpose we can use other methods such as: -Chi-Squared Test for categorical features -Spearman's Rank Correlation: Tests whether two samples have a monotonic relationship.

However, we can draw some other insights about the data -People of age above 23 are less ikely to subscrbe the term deposit.

In [77]:
```
#Hypothesis Test to prove: "People of age above 23 are less likely to subscrb
#create a new column called is_old and fill with true
df_bank_full['is_middle_aged'] = False
df_bank_full.loc[df_bank_full['age'] <=23 , 'is_middle_aged'] = True
```

In [78]:
```
df_bank_full['y'].replace(['no', 'yes'], [0,1], inplace = True)
```

In [79]:
```
X= df_bank_full.groupby('is_middle_aged')['y'].mean().sort_values().plot(kind
```



In [80]:
```
df_bank_full['y'].replace([0, 1], ['no','yes'], inplace = True)
```

In [81]:
```
# df_bank_full[['job','y']]
```

From the above plot we can say our hypothesis was correct.

In [82]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df_bank_full.corr(), annot=True,linewidths=.5, ax=ax)
plt.show()
```

From the above correlation matrix, there seems to be strong correlation between the features p_days and previous. we can verify that using hypothesis test.

In [83]:
```
#Hypothesis test to check if there is a relationship between p_days and previ
from scipy.stats import pearsonr
stat, p = pearsonr(df_bank_full['pdays'], df_bank_full['previous'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=0.454, p=0.000
Probably dependent
```

In [84]:
```
from scipy.stats import spearmanr
stat, p = spearmanr(df_bank_full['pdays'], df_bank_full['previous'])
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=0.986, p=0.000
Probably dependent
```

From the above hypothesis we can conclude that the two might be related

In [85]:
```
# Hypothesis test to check if there is a correlation between our target varia
```

In [86]:
```
# y and jobs
chisqt = pd.crosstab(df_bank_full.y, df_bank_full.job, margins=True)
print(chisqt)
```

```
job  admin.  blue-collar  entrepreneur  housemaid  management  retired  \
y
no     4482         8986          1352       1134        8291     1790
yes     689          749           135        157        1167      504
All    5171         9735          1487       1291        9458     2294

job  self-employed  services  student  technician  unemployed    All
y
no            1417      3776      786        6765        1143  39922
yes            162       378      356         832         160   5289
All           1579      4154     1142        7597        1303  45211
```

In [87]:

```
from scipy.stats import chi2_contingency
stat, p, dof, expected = chi2_contingency(chisqt)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=866.722, p=0.000
Probably dependent
```

In [88]:

```
# y and education
chisqt_edu = pd.crosstab(df_bank_full.y, df_bank_full.education, margins=True
stat, p, dof, expected = chi2_contingency(chisqt_edu)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=109.759, p=0.000
Probably dependent
```

In [89]:

```
# y and marital
chisqt_mar = pd.crosstab(df_bank_full.y, df_bank_full.marital, margins=True)
stat, p, dof, expected = chi2_contingency(chisqt_mar)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=196.496, p=0.000
Probably dependent
```

In [90]:

```
# y and default
chisqt_def = pd.crosstab(df_bank_full.y, df_bank_full.default, margins=True)
stat, p, dof, expected = chi2_contingency(chisqt_def)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=22.724, p=0.000
Probably dependent
```

In [91]:

```
# y and housing
chisqt_hou = pd.crosstab(df_bank_full.y, df_bank_full.housing, margins=True)
stat, p, dof, expected = chi2_contingency(chisqt_hou)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=875.694, p=0.000
Probably dependent
```

In [92]:

```
# y and loan
chisqt_loa = pd.crosstab(df_bank_full.y, df_bank_full.loan, margins=True)
stat, p, dof, expected = chi2_contingency(chisqt_loa)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

```
stat=210.195, p=0.000
Probably dependent
```

In [93]:

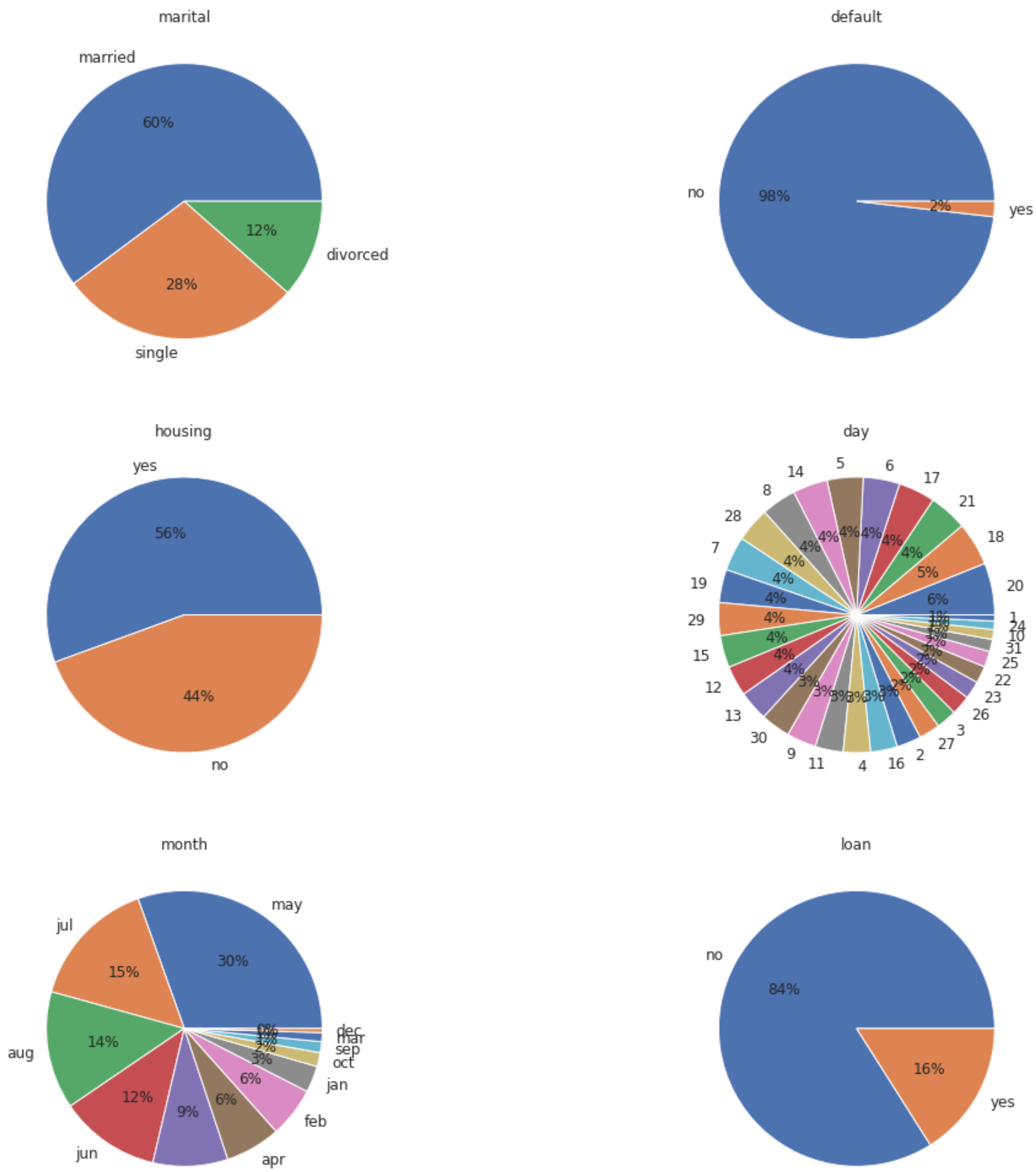
```
# y and day
chisqt_day = pd.crosstab(df_bank_full.y, df_bank_full.day, margins=True)
stat, p, dof, expected = chi2_contingency(chisqt_day)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
        print('Probably independent')
else:
        print('Probably dependent')
```

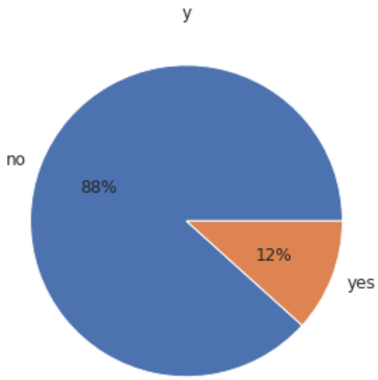
```
stat=574.051, p=0.000
Probably dependent
```

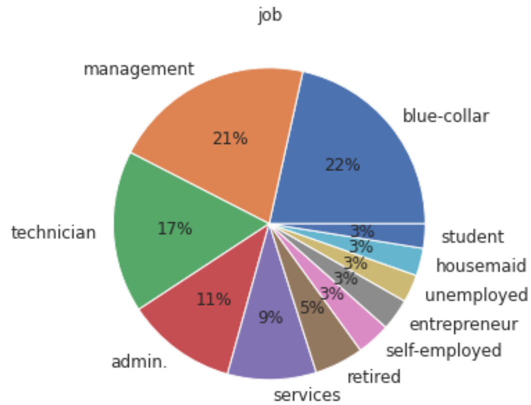
Univariate Analysis on Categorical data

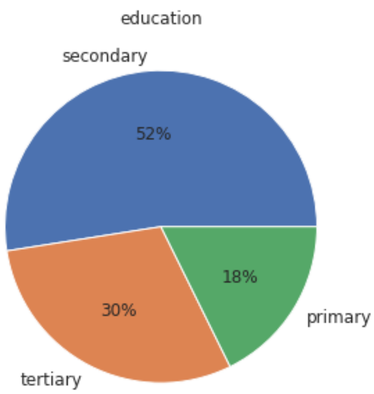
In [94]:

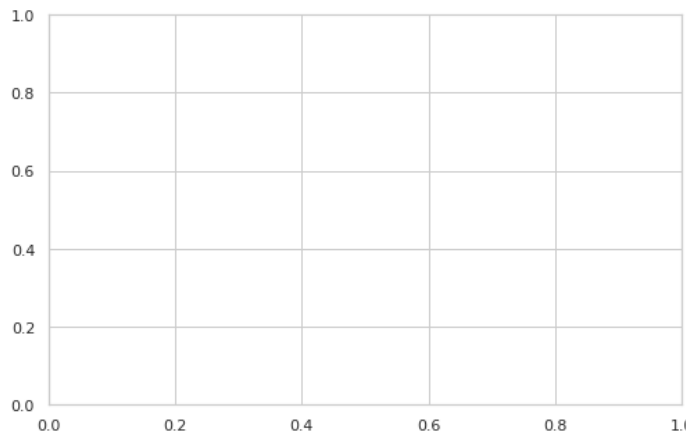
```
#1. Pie chart to see propotion of samples
cat_cols=['marital', 'default','housing','day', 'month','loan','y','job','edu
fig, axes = plt.subplots(5, 2, figsize=(18,30))
axes = [ax for axes_rows in axes for ax in axes_rows]

for i, c in enumerate(df_bank_full[cat_cols]):
    df_bank_full[c].value_counts().plot(kind='pie',
                                         ax=axes[i],
                                         title=c,
                                         autopct='%.0f%%',
                                         fontsize=12)
    axes[i].set_ylabel('')
```





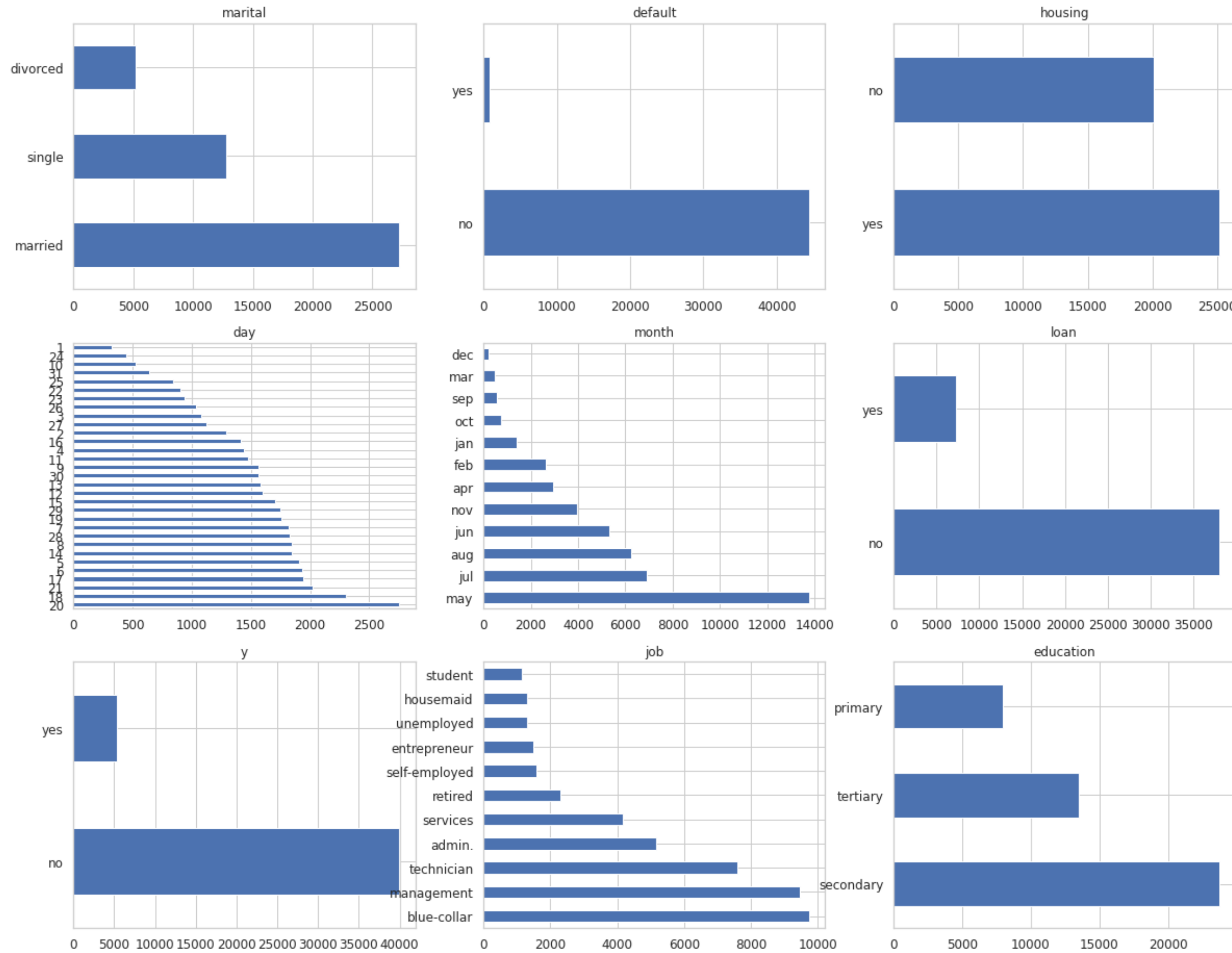






In [95]:

```
#2. Bar plot to see frequency

fig, axes = plt.subplots(3, 3, figsize=(20,16))
axes = [ax for axes_rows in axes for ax in axes_rows]

for i, c in enumerate(df_bank_full[cat_cols]):
    df_bank_full[c].value_counts().plot(kind='barh',
                                        ax=axes[i],
                                        title=c,
                                        fontsize=12)
```



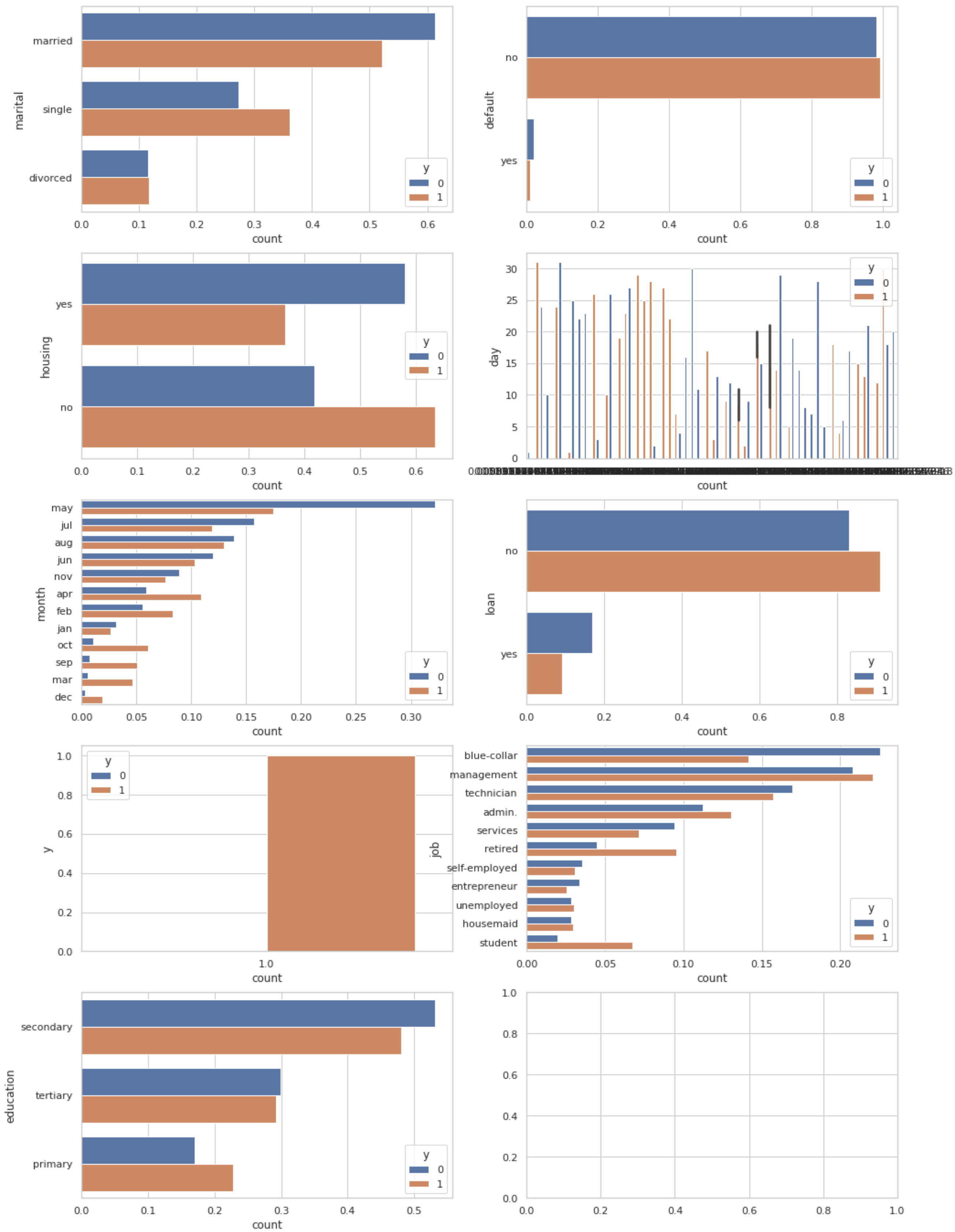
Observations

Less number of students and more number of management and technician customers Most of married customers, Most customers education levels is secondary, Most cutomers are not defaulted in past, More than 50% have taken housing loan, Nearly 85% have taken personal loan, Major communication type is cellular, Most of the customers were last contacted in the month of May, Most customers where not contacted in previous month

Bivariate Analysis for Categorical data

In [100…

```
target_col='y'
train=df_bank_full
train['y'].replace(['no', 'yes'], [0,1], inplace = True)
fig, axes = plt.subplots(5, 2, figsize=(16,24))
axes = [ax for axes_rows in axes for ax in axes_rows]

for i, c in enumerate(train[cat_cols]):
    #index of rows where target_col value is 0
    fltr = train[target_col]==0

    #dataframe conraining rows and columns where target_col value is 0
    #fltr-index of rows where target_col value is 0
    #c-column name
    #taking the value count
    #resetting index as column name
    vc_a=train[fltr][c].value_counts(normalize=True).reset_index().rename({'i

    #dataframe conraining rows and columns where target_col value is 1
    vc_b=train[~fltr][c].value_counts(normalize=True).reset_index().rename({'

    #setting target_col value to 0 and 1 respectively
    vc_a[target_col]=0
    vc_b[target_col]=1

    #combining into single dataframe
    df = pd.concat([vc_a, vc_b]).reset_index(drop=True)

    #plotting
    sns.barplot(y=c, x='count', data=df, hue='y', ax=axes[i])
```



married
single
divorced
marital
count
y
0
1
no
yes
default
housing
day
month
may
jul
aug
jun
nov
apr
feb
jan
oct
sep
mar
dec
loan
job
blue-collar
management
technician
admin.
services
retired
self-employed
entrepreneur
unemployed
housemaid
student
education
secondary
tertiary
primary

Observations

Management, retire, self-employed, unemployed and students tend to subscribe more, Singles subscribe more than married and divorced, Customers with tertiary level of education will subscribe, Customers with without housing and personal load tend to subscribe to team deposit, Customers approached by cellular communication have subscribed, Subscription rate is more during start(jan,feb,march,apr) and end of the year(oct,sept,dec), Customers who subscribed during previous campaign tend to subscribe more.

In [ ]:

```

```