In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Asus\Downloads\DsResearch (1)\DsResearch
```

Case Study of Banking Data

The dataset is about detailed records of 45,211 customers of a Portuguese bank, ranging between May 2008 and November 2010. The primary goal is to predict whether a customer will subscribe to a term deposit product.

Column Description

- `age` : Age of the bank's clients. It has numeric values indicating age in year.

- `job` : Client's job description.

- `marital` : Marital status of client.

- `education` : level of education of client.

- `default` : This column indicates whether the client has credit in default. It's a binary variable with options "yes" or "no"

- `balance` : Average yearly balance in client's account.

- `housing` : Indicates whether client has a house loan. Binary variable with values "yes" or "no".

- `loan` : Indicates whether client has a house loan. Is a binary variable with values "yes" or "no".

- `contact` : Type of communication with client.

- `day` : Last contact day of the month.

- `month` : Last contact month of the year.

- `duration` : Duration of last contact in seconds.

- `campaign` : number of contacts performed during the campaign.

- `pdays` : Number of days that passed by after the client was last contacted. numeric value where -1 means client was not contacted previously.

- `previous` : Number of contacts performed before this campaign for this client.

- `poutcome` : Outcome of the previous marketing campaign.

Cleaning Data

In [3]:
```
df1 = df.copy()
df.head()
```

Out[3]:

	age	job	marital	marital_status	education	default	balance	housing
0	58	management	married	married	tertiary	no	2143	yes
1	44	technician	single	single	secondary	no	29	yes
2	33	entrepreneur	married	married	secondary	no	2	yes
3	47	blue-collar	married	married	unknown	no	1506	yes
4	33	unknown	single	single	unknown	no	1	no

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
age               0
job               0
marital           3
marital_status    3
education         3
default           0
balance           0
housing           0
loan              0
contact           0
day               0
month             0
day_month         0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
y                 0
dtype: int64
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45216 entries, 0 to 45215
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             45216 non-null  int64 
 1   job             45216 non-null  object
 2   marital         45213 non-null  object
 3   marital_status  45213 non-null  object
 4   education       45213 non-null  object
 5   default         45216 non-null  object
 6   balance         45216 non-null  int64 
 7   housing         45216 non-null  object
 8   loan            45216 non-null  object
 9   contact         45216 non-null  object
 10  day             45216 non-null  int64 
 11  month           45216 non-null  object
 12  day_month       45216 non-null  object
 13  duration        45216 non-null  int64 
 14  campaign        45216 non-null  int64 
 15  pdays           45216 non-null  int64 
 16  previous        45216 non-null  int64 
 17  poutcome        45216 non-null  object
 18  y               45216 non-null  object
dtypes: int64(7), object(12)
memory usage: 6.6+ MB
```

In [6]:
```
df1.describe()
```

Out[6]:

	age	balance	day	duration	campaign	
count	45216.000000	45216.000000	45216.000000	45216.000000	45216.000000	452
mean	40.938186	1362.277844	15.806507	258.166202	2.763668	
std	10.621249	3044.609674	8.322022	257.515482	3.097896	1
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	
25%	33.000000	72.000000	8.000000	103.000000	1.000000	
50%	39.000000	448.500000	16.000000	180.000000	2.000000	
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	8

In [7]:
```
df1['education'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 45216 entries, 0 to 45215
Series name: education
Non-Null Count  Dtype 
--------------  ----- 
45213 non-null  object
dtypes: object(1)
memory usage: 353.4+ KB
```

In [8]:
```
df1[df1.duplicated()]
```

Out[8]:

	age	job	marital	marital_status	education	default	balance	hous
45211	29	management	single	single	tertiary	no	765	
45212	68	retired	married	married	secondary	no	1146	
45213	53	management	married	married	tertiary	no	583	
45214	73	retired	married	married	secondary	no	2850	
45215	71	retired	divorced	divorced	primary	no	1729	

In [9]:
```
df1['job'].value_counts()
```

Out[9]:
```
blue-collar      9732
management       9460
technician       7597
admin.           5171
services         4154
retired          2267
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: job, dtype: int64
```

In [10]:
```
df1['marital_status'].value_counts()
```

Out[10]:
```
married     27216
single      12790
divorced     5207
Name: marital_status, dtype: int64
```

In [11]:
```
df1['default'].value_counts()
```

Out[11]:
```
no     44401
yes      815
Name: default, dtype: int64
```

In [12]:
```
df1['education'].value_counts()
```

Out[12]:
```
secondary    23204
tertiary     13301
primary       6851
unknown       1857
Name: education, dtype: int64
```

In [13]:
```
df1['housing'].value_counts()
```

Out[13]:
```
yes    25130
no     20086
Name: housing, dtype: int64
```

In [14]:
```
df1['loan'].value_counts()
```

Out[14]:
```
no     37972
yes     7244
Name: loan, dtype: int64
```

In [15]:
```
df1['contact'].value_counts()
```

Out[15]:
```
cellular     29290
unknown      13020
telephone     2906
Name: contact, dtype: int64
```

In [16]:
```
df1['poutcome'].value_counts()
```

Out[16]:
```
unknown    36961
failure     4902
other       1840
success     1513
Name: poutcome, dtype: int64
```

In [17]:
```
df1['y'].value_counts()
```

Out[17]:
```
no     39922
yes     5294
Name: y, dtype: int64
```

In [18]:
```
# Since there is no way to fill the missing data,
# replacing those with string 'No Data'

df1 = df1.fillna('No Data')
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45216 entries, 0 to 45215
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             45216 non-null  int64 
 1   job             45216 non-null  object
 2   marital         45216 non-null  object
 3   marital_status  45216 non-null  object
 4   education       45216 non-null  object
 5   default         45216 non-null  object
 6   balance         45216 non-null  int64 
 7   housing         45216 non-null  object
 8   loan            45216 non-null  object
 9   contact         45216 non-null  object
 10  day             45216 non-null  int64 
 11  month           45216 non-null  object
 12  day_month       45216 non-null  object
 13  duration        45216 non-null  int64 
 14  campaign        45216 non-null  int64 
 15  pdays           45216 non-null  int64 
 16  previous        45216 non-null  int64 
 17  poutcome        45216 non-null  object
 18  y               45216 non-null  object
dtypes: int64(7), object(12)
memory usage: 6.6+ MB
```

In [19]:
```
# missing year column
month_dict = {month: 1 if month in ['jan', 'feb', 'mar'] else 0 for
print(month_dict)
```

```
{'jan': 1, 'feb': 1, 'mar': 1, 'apr': 0, 'may': 0, 'jun': 0, 'ju
l': 0, 'aug': 0, 'sep': 0, 'oct': 0, 'nov': 0, 'dec': 0}
```

In [20]:
```
df1 = df1.reset_index()
df.columns
```

Out[20]:
```
Index(['age', 'job', 'marital', 'marital_status', 'education', 'de
fault',
       'balance', 'housing', 'loan', 'contact', 'day', 'month', 'd
ay_month',
       'duration', 'campaign', 'pdays', 'previous', 'poutcome',
'y'],
      dtype='object')
```

In [21]:
```
def find_year(index):
    month = df1.iloc[index,12]
    if index != 0 :
        month_prev = df1.iloc[index-1,12]
        if(month == month_prev):
            return 2008 + month_dict[month]
        else :
            month_dict[month_prev] = month_dict[month_prev] + 1
            return 2008 + month_dict[month]
    else : return 2008
```

In [22]:
```
df1['year'] = df1['index'].apply(find_year)
df1['year'].value_counts()
```

Out[22]:
```
2008    30729
2009    12373
2010     2114
Name: year, dtype: int64
```

In [23]:
```
df1['year'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 45216 entries, 0 to 45215
Series name: year
Non-Null Count  Dtype
--------------  -----
45216 non-null  int64
dtypes: int64(1)
memory usage: 353.4 KB
```

In [24]:
```
# changing dtype to categorical for appropriate columns

df1[['job','marital','marital_status','education','default','housing
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45216 entries, 0 to 45215
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           45216 non-null  int64
 1   age             45216 non-null  int64
 2   job             45216 non-null  category
 3   marital         45216 non-null  category
 4   marital_status  45216 non-null  category
 5   education       45216 non-null  category
 6   default         45216 non-null  category
 7   balance         45216 non-null  int64
 8   housing         45216 non-null  category
 9   loan            45216 non-null  category
 10  contact         45216 non-null  category
 11  day             45216 non-null  int64
 12  month           45216 non-null  object
 13  day_month       45216 non-null  object
 14  duration        45216 non-null  int64
 15  campaign        45216 non-null  int64
 16  pdays           45216 non-null  int64
 17  previous        45216 non-null  int64
 18  poutcome        45216 non-null  category
 19  y               45216 non-null  category
 20  year            45216 non-null  int64
dtypes: category(10), int64(9), object(2)
memory usage: 4.2+ MB
```

In [25]:
```
# marital and marital_columns are exactly the same, therefore
# removing one and keeping just 'marital_status' column

df1 = df1.drop('marital', axis=1)
```

In [26]:
```
# merging day, month, day_month and year column into one

dt_sample = pd.DatetimeIndex([dt.datetime(2023,1,1),dt.datetime(2022
type(dt_sample)
```

Out[26]:
```
pandas.core.indexes.datetimes.DatetimeIndex
```

In [27]:
```
df1.columns
```

Out[27]:
```
Index(['index', 'age', 'job', 'marital_status', 'education', 'defa
ult',
       'balance', 'housing', 'loan', 'contact', 'day', 'month', 'd
ay_month',
       'duration', 'campaign', 'pdays', 'previous', 'poutcome',
'y', 'year'],
      dtype='object')
```

In [28]:
```
df1['day'] = df1['day'].astype(str).str.zfill(2)
df1['year'] = df1['year'].astype(str)
df1['date'] = df1['day'] + '-' + df1['month'] + '-' + df1['year']
df1['date'] = pd.to_datetime(df1['date'], format='%d-%b-%Y')
df1['date']
```

Out[28]:
```
0        2008-05-05
1        2008-05-05
2        2008-05-05
3        2008-05-05
4        2008-05-05
            ...
45211    2010-11-16
45212    2010-11-16
45213    2010-11-17
45214    2010-11-17
45215    2010-11-17
Name: date, Length: 45216, dtype: datetime64[ns]
```

In [29]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45216 entries, 0 to 45215
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           45216 non-null  int64
 1   age             45216 non-null  int64
 2   job             45216 non-null  category
 3   marital_status  45216 non-null  category
 4   education       45216 non-null  category
 5   default         45216 non-null  category
 6   balance         45216 non-null  int64
 7   housing         45216 non-null  category
 8   loan            45216 non-null  category
 9   contact         45216 non-null  category
 10  day             45216 non-null  object
 11  month           45216 non-null  object
 12  day_month       45216 non-null  object
 13  duration        45216 non-null  int64
 14  campaign        45216 non-null  int64
 15  pdays           45216 non-null  int64
 16  previous        45216 non-null  int64
 17  poutcome        45216 non-null  category
 18  y               45216 non-null  category
 19  year            45216 non-null  object
 20  date            45216 non-null  datetime64[ns]
dtypes: category(9), datetime64[ns](1), int64(7), object(4)
memory usage: 4.5+ MB
```

In [30]:
```
df1 = df1.drop(['day', 'month', 'day_month'], axis=1)
```

In [31]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45216 entries, 0 to 45215
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           45216 non-null  int64
 1   age             45216 non-null  int64
 2   job             45216 non-null  category
 3   marital_status  45216 non-null  category
 4   education       45216 non-null  category
 5   default         45216 non-null  category
 6   balance         45216 non-null  int64
 7   housing         45216 non-null  category
 8   loan            45216 non-null  category
 9   contact         45216 non-null  category
 10  duration        45216 non-null  int64
 11  campaign        45216 non-null  int64
 12  pdays           45216 non-null  int64
 13  previous        45216 non-null  int64
 14  poutcome        45216 non-null  category
 15  y               45216 non-null  category
 16  year            45216 non-null  object
 17  date            45216 non-null  datetime64[ns]
dtypes: category(9), datetime64[ns](1), int64(7), object(1)
memory usage: 3.5+ MB
```

In [32]:
```
# Dropping Duplicates
df1[df1.duplicated(subset=[ 'age', 'job', 'marital_status', 'educati
       'balance', 'housing', 'loan', 'contact', 'date', 'duration',
       'y', 'year'],keep='first')]
```

Out[32]:

	index	age	job	marital_status	education	default	balance	housin
45211	45211	29	management	single	tertiary	no	765	n
45212	45212	68	retired	married	secondary	no	1146	n
45213	45213	53	management	married	tertiary	no	583	n
45214	45214	73	retired	married	secondary	no	2850	n
45215	45215	71	retired	divorced	primary	no	1729	n

In [33]:
```
df1 = df1.drop_duplicates(subset=[ 'age', 'job', 'marital_status', '
       'balance', 'housing', 'loan', 'contact', 'date', 'duration',
       'y', 'year'],keep='first')
```

In [34]:
```
df1[df1.duplicated(subset=[ 'age', 'job', 'marital_status', 'educati
       'balance', 'housing', 'loan', 'contact', 'date', 'duration',
       'y', 'year'],keep='first')]
```

Out[34]:

index	age	job	marital_status	education	default	balance	housing	loan	conta

In [35]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45211 entries, 0 to 45210
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           45211 non-null  int64
 1   age             45211 non-null  int64
 2   job             45211 non-null  category
 3   marital_status  45211 non-null  category
 4   education       45211 non-null  category
 5   default         45211 non-null  category
 6   balance         45211 non-null  int64
 7   housing         45211 non-null  category
 8   loan            45211 non-null  category
 9   contact         45211 non-null  category
 10  duration        45211 non-null  int64
 11  campaign        45211 non-null  int64
 12  pdays           45211 non-null  int64
 13  previous        45211 non-null  int64
 14  poutcome        45211 non-null  category
 15  y               45211 non-null  category
 16  year            45211 non-null  object
 17  date            45211 non-null  datetime64[ns]
dtypes: category(9), datetime64[ns](1), int64(7), object(1)
memory usage: 3.8+ MB
```

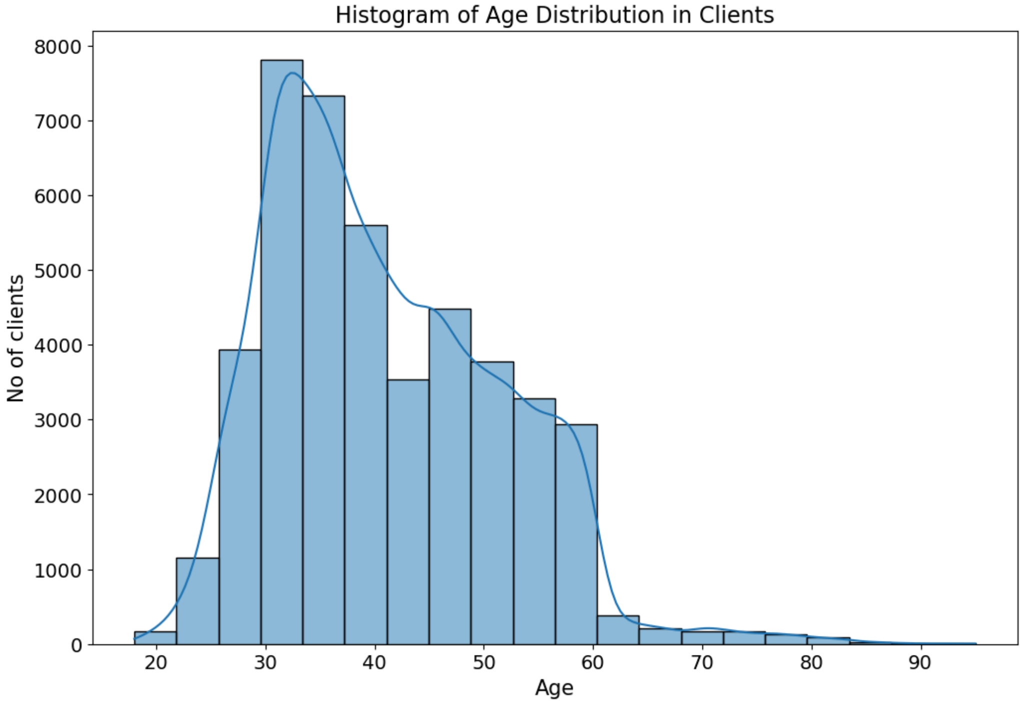
Exploratory Data Analysis

1. Distribution of age among clients

In [36]:
```
df1['age'].describe()
```

Out[36]:
```
count    45211.000000
mean        40.936210
std         10.618762
min         18.000000
25%         33.000000
50%         39.000000
75%         48.000000
max         95.000000
Name: age, dtype: float64
```

In [37]:
```
plt.figure(figsize=(12,8))
sns.histplot(df1['age'],bins=20,kde=True)
plt.title('Histogram of Age Distribution in Clients',fontsize = 16)
plt.xlabel('Age',fontsize=15)
plt.ylabel('No of clients',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



In [38]:
```
df1[(df1['age']>=30) & (df1['age']<=40)].shape[0]/df1.shape[0]
```

Out[38]: 0.4300723275309106

In [39]:
```
df1[df1['age']>70].shape[0]
```

Out[39]: 487

Conclusions:

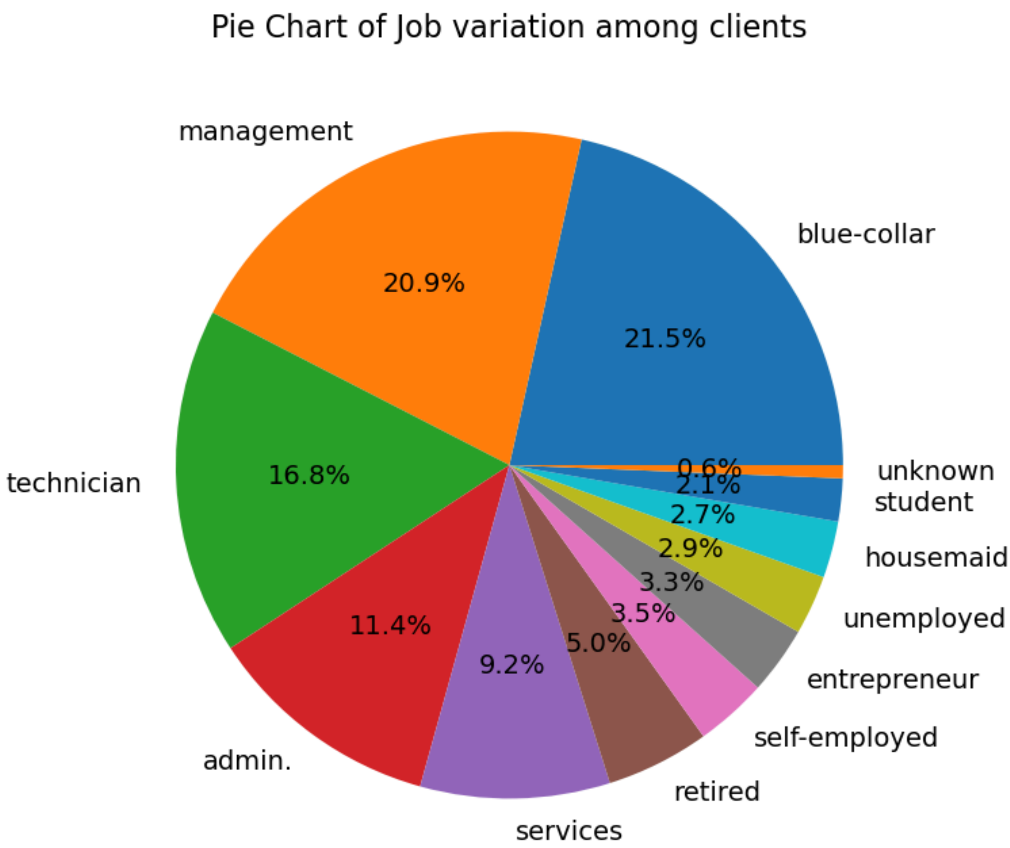
1. 43% of the clients are between the age of 30 and 40
2. Clients above the age of 70 are classified as outliers (487 such entries)
3. The median is 39 years

2. Job type variation among clients

In [40]:
```
df1['job'].value_counts()
```

Out[40]:
```
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: job, dtype: int64
```

In [41]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['job'].value_counts().tolist(),labels=df1['job'].value_c
plt.title('Pie Chart of Job variation among clients',fontsize=16)
plt.show()
```



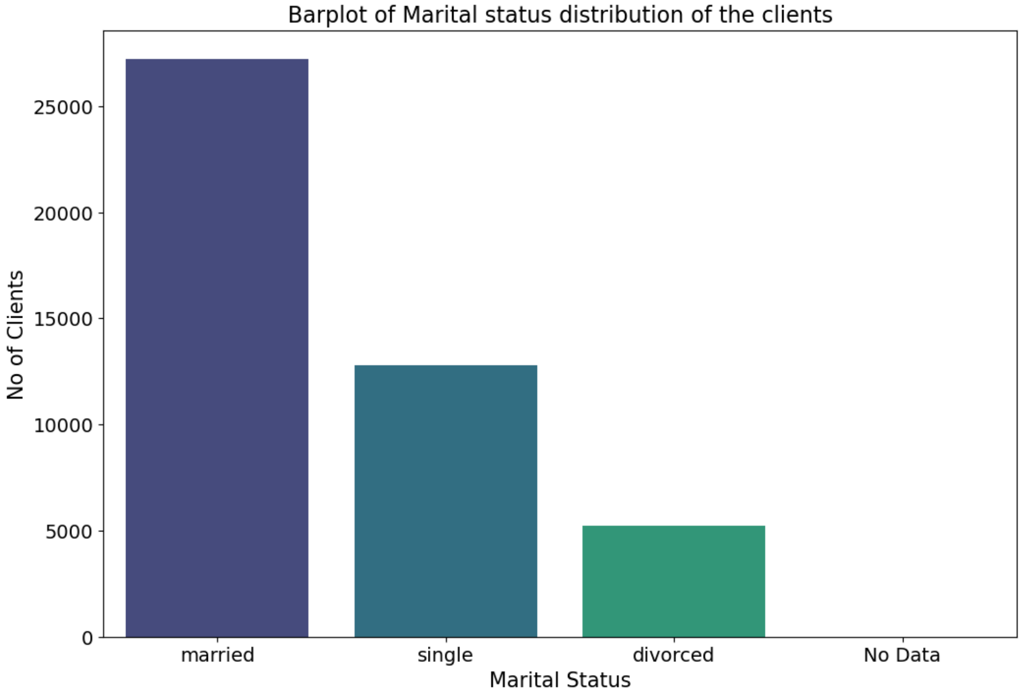

```
Conclusion:
```

1. Majority of the clients(42.4%) have blue collar jobs or are in some management role.
2. Only 2.1% of the clients are students which is very less.

3. There are relatively fewer clients who are self-employed, entrepreneurs, unemployed, housemaids, and students.
4. The 'student' and 'unknown' categories have the smallest number of clients

3. Marital status distribution among clients

In [42]:
```
order = ['married', 'single', 'divorced', 'No Data']
plt.figure(figsize=(12,8))
sns.barplot(data=df1,x=df1['marital_status'].value_counts(sort=True)
#plt.xticks(rotation=45)
plt.title('Barplot of Marital status distribution of the clients',fo
plt.xlabel('Marital Status',fontsize=15)
plt.ylabel('No of Clients',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

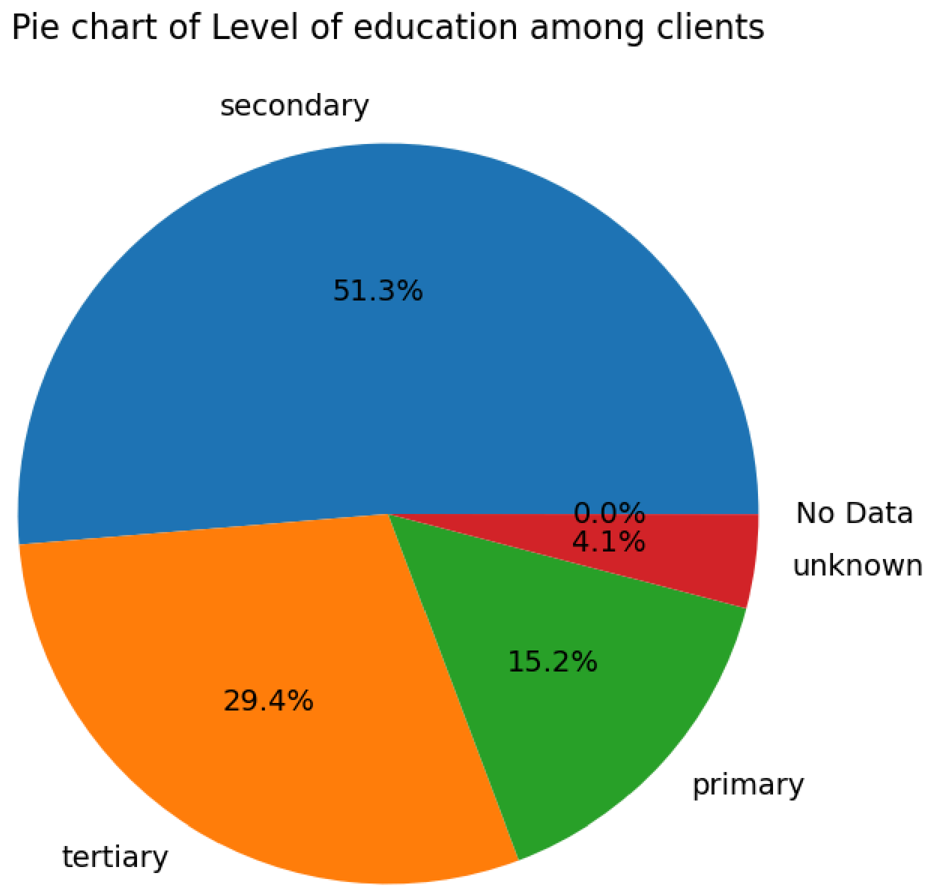


Conclusion:

1. Majority of the clients are Married.
2. Single clients are the next most common group but are less than half the number of married clients.
3. Divorced clients represent a smaller fraction compared to the married and single clients.
4. There is a small category labeled "No Data", indicating that there are some clients for whom the marital status is not recorded.

Level of Education among clients

In [43]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['education'].value_counts().tolist(),labels=df1['educati
plt.title('Pie chart of Level of education among clients',fontsize=1
plt.show()
```

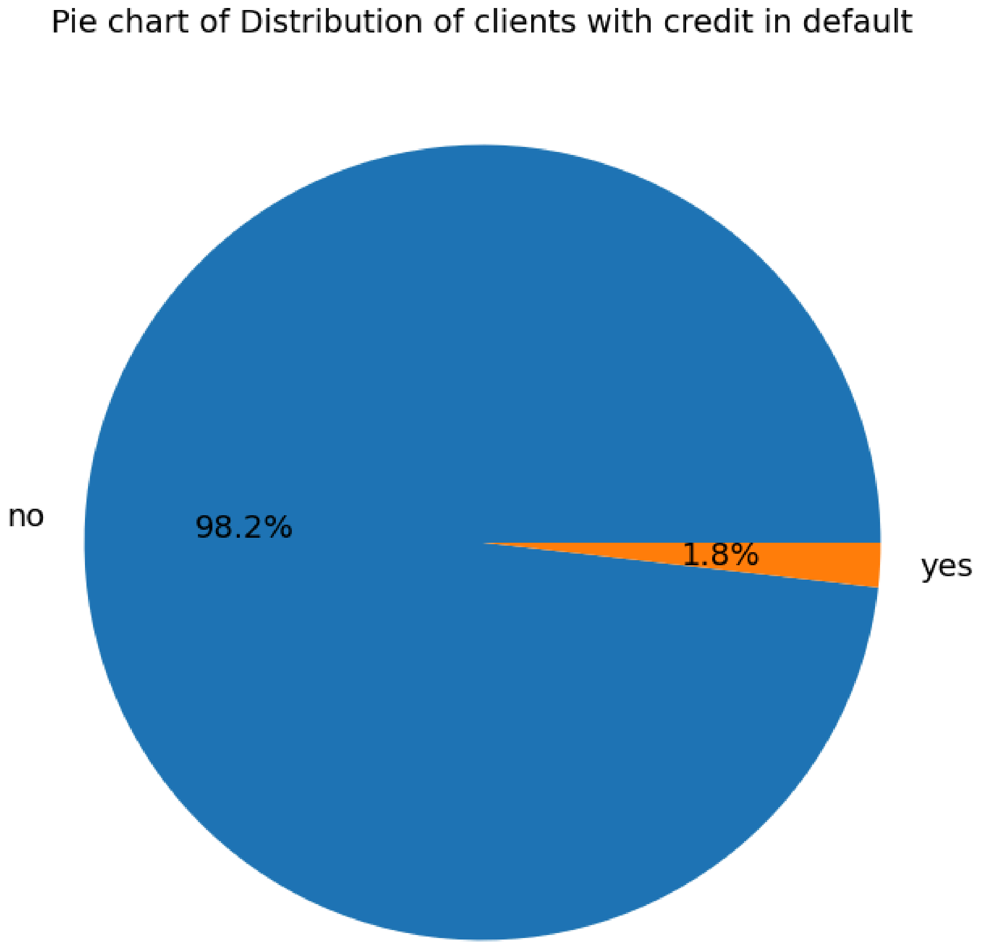


Conclusions:

1. Majority of the clients(51.3%) have completed their secondary education.
2. The next substantial group consists of clients with tertiary education, indicating a significant number of clients with higher education.
3. Clients with primary education form a smaller proportion compared to the other two educational levels.
4. There is a category of clients for whom the level of education is unknown.
5. A small fraction of the data does not have education level information, indicated as "No Data".

5. Proportion of clients that have credit in default

In [44]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['default'].value_counts().tolist(),labels=df1['default']
plt.title('Pie chart of Distribution of clients with credit in defau
plt.show()
```



Conclusion:

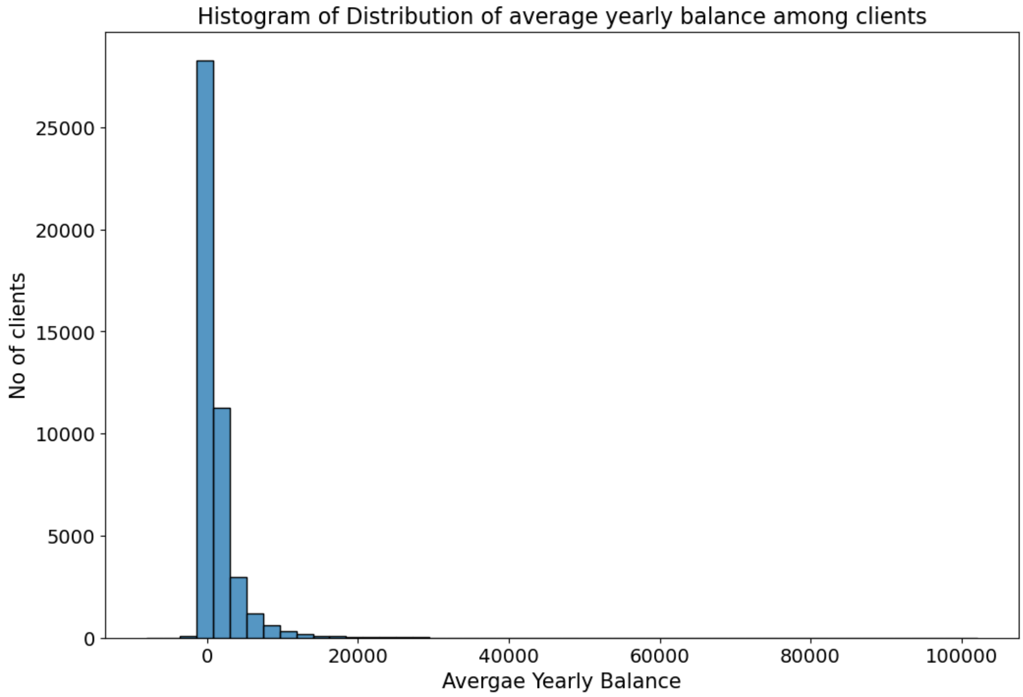
1. Only 1.8%(815) of the clients have credit in default

6. Distribution of average yearly balance among clients

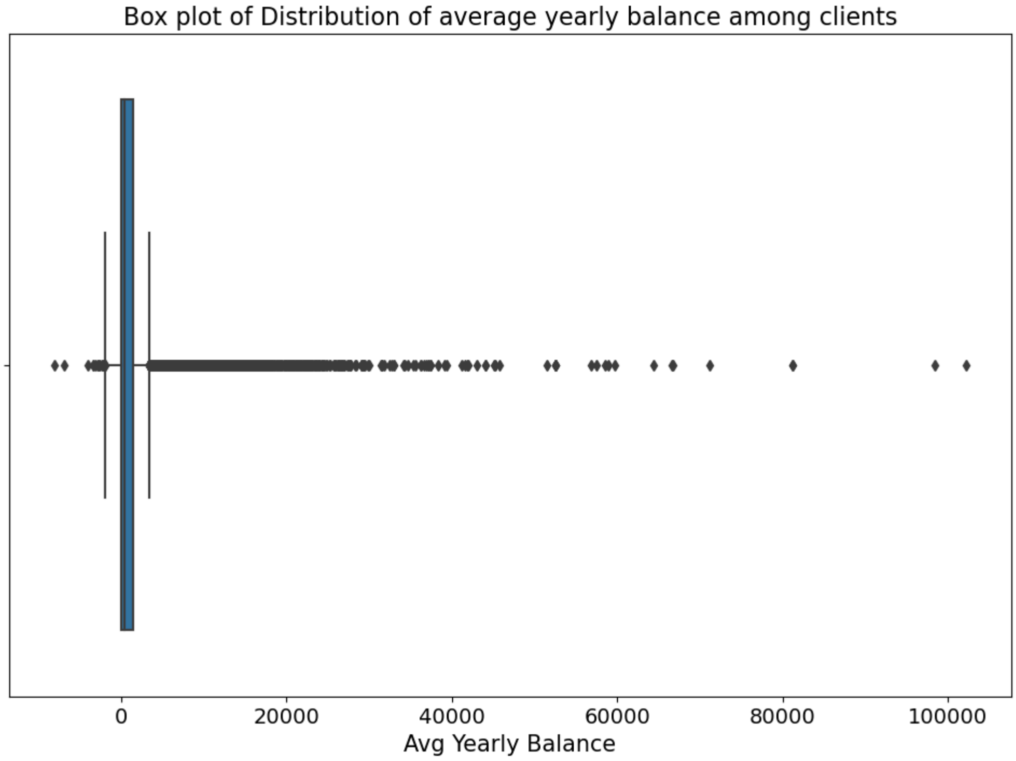
In [45]:
```
df1['balance'].describe()
```

Out[45]:
```
count     45211.000000
mean       1362.272058
std        3044.765829
min       -8019.000000
25%          72.000000
50%         448.000000
75%        1428.000000
max      102127.000000
Name: balance, dtype: float64
```

In [46]:
```
plt.figure(figsize=(12,8))
sns.histplot(df1['balance'],bins=50)
plt.title('Histogram of Distribution of average yearly balance among
plt.xlabel('Avergae Yearly Balance',fontsize=15)
plt.ylabel('No of clients',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
# plt.xlim(-5000,35000)
plt.show()
```



In [47]:
```
plt.figure(figsize=(12,8))
sns.boxplot(data=df1,x='balance')
plt.title('Box plot of Distribution of average yearly balance among
plt.xlabel('Avg Yearly Balance',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

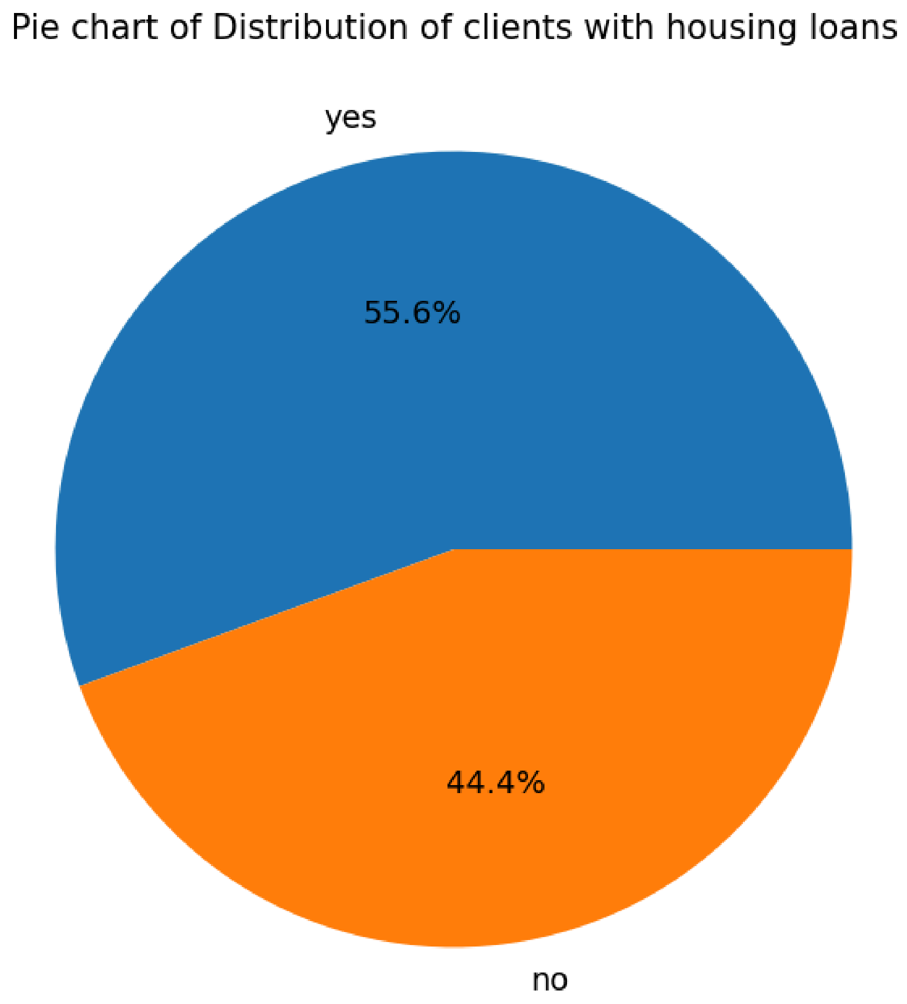


Conclusions:

1. A large majority of clients have a relatively low average yearly balance, as indicated by the tall bar at the beginning of the histogram.
2. The frequency of clients decreases rapidly as the balance amount increases, suggesting that higher balances are much less common.
3. There are very few clients with an average yearly balance above 20,000 euros, indicating that high balances are rare within this client base.
4. The distribution is right-skewed, with most clients clustered in the lower balance range and outliers with high balances.
5. Considering the shape of the distribution, the bank’s client base is likely comprised of individuals with modest means rather than high-net-worth individuals.
6. The median balance 448 which is relatively low, suggesting that the typical client does not have a large average yearly balance.

7. Clients with housing loan

In [48]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['housing'].value_counts().tolist(),labels=df1['housing']
plt.title('Pie chart of Distribution of clients with housing loans',
plt.show()
```

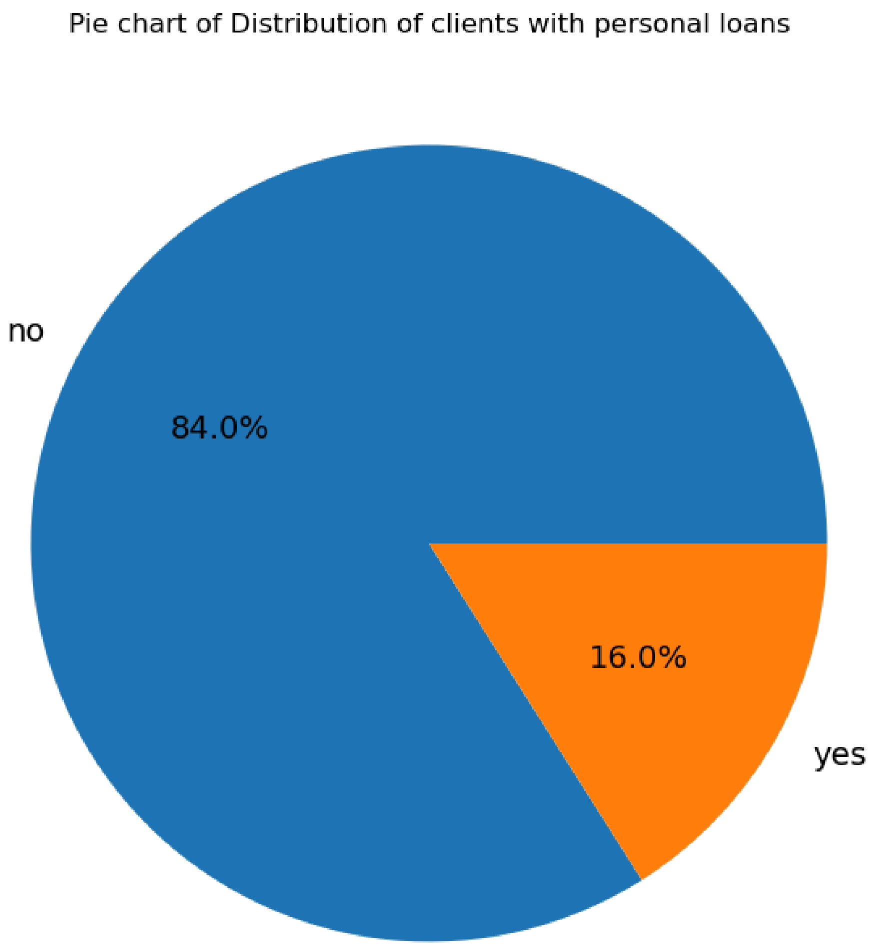


Conclusion:

- Majority of the clients (55.6%) have housing loans

8. Clients with personal loans

In [49]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['loan'].value_counts().tolist(),labels=df1['loan'].value
plt.title('Pie chart of Distribution of clients with personal loans'
plt.show()
```

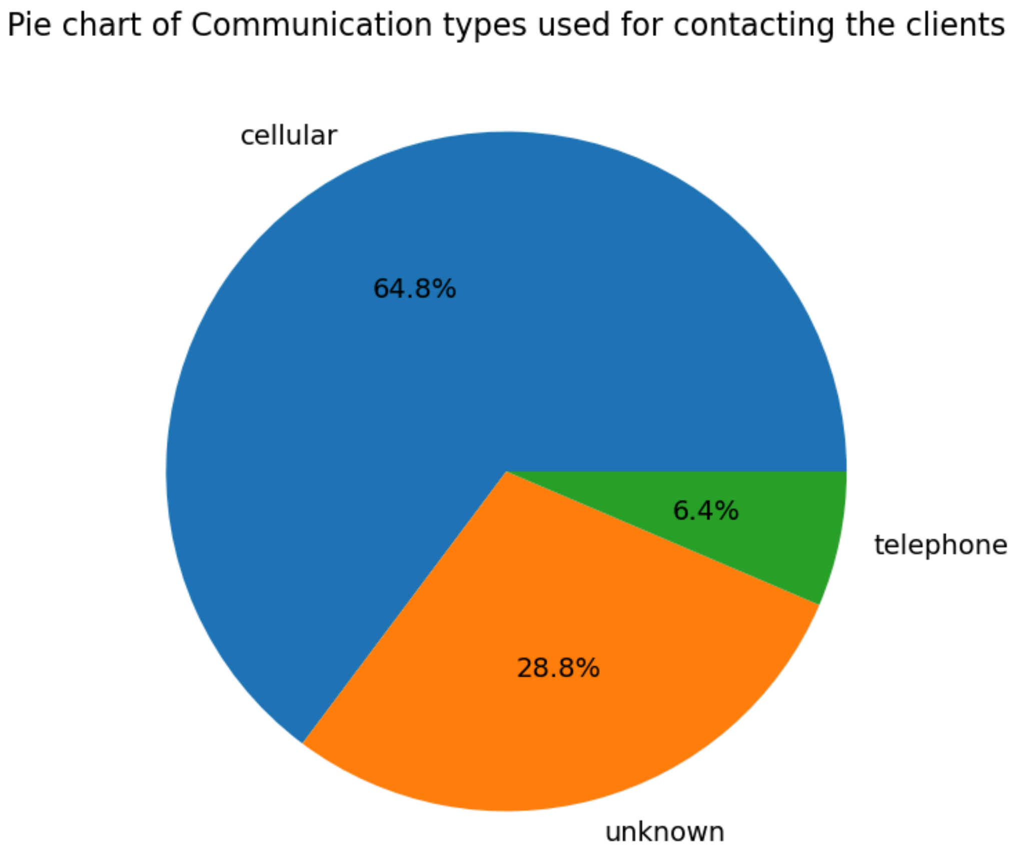


Conclusion:

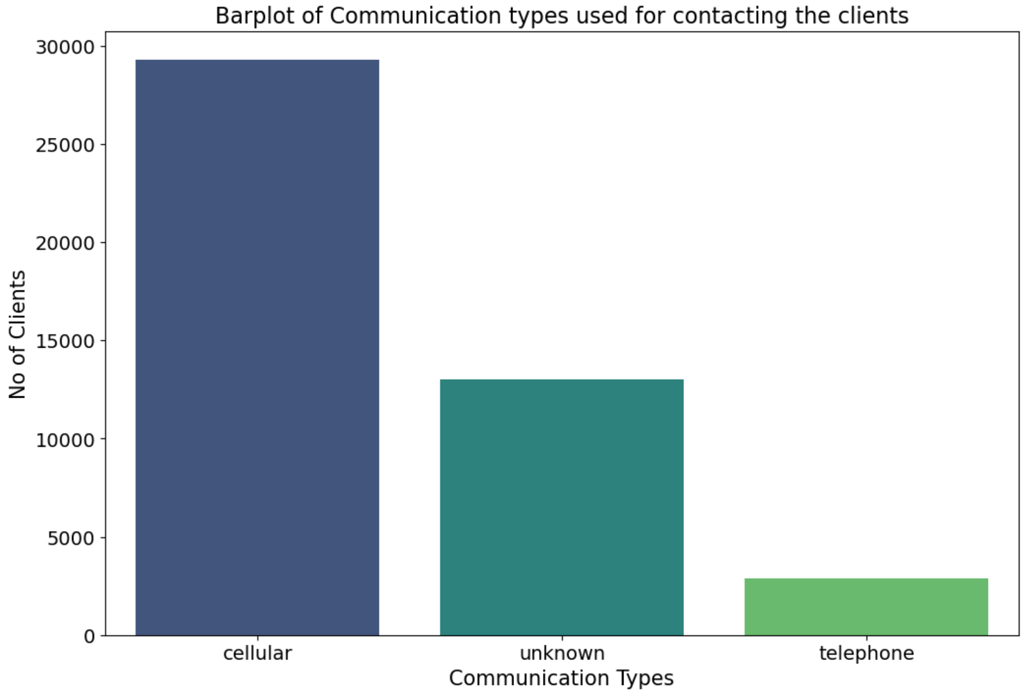
- Majority of the clients (84%) of the clients don't have any personal loans

9. Communication types used for contacting clients during the campaign

In [50]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['contact'].value_counts().tolist(),labels=df1['contact']
plt.title('Pie chart of Communication types used for contacting the
plt.show()
```



In [51]:
```
plt.figure(figsize=(12,8))
sns.barplot(data=df1,x=df1['contact'].value_counts(sort=True).keys()
#plt.xticks(rotation=45)
plt.title('Barplot of Communication types used for contacting the cl
plt.xlabel('Communication Types',fontsize=15)
plt.ylabel('No of Clients',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

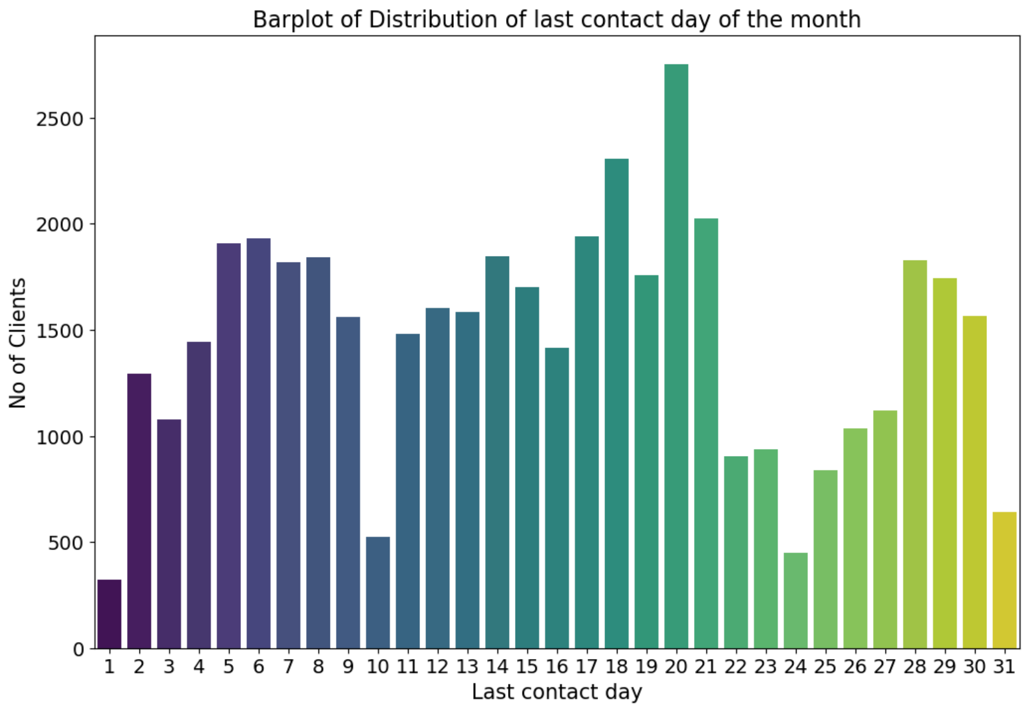


Conclusions:

1. 64.8 % pf the clients were contacted using a cellular medium.
2. only 6.4 % of the clients were contacted using telephone.
3. A very large percentage of the clients (28.8%) were contacted using unknown means.

10. Distribution of the last contact day of the month

In [52]:
```
plt.figure(figsize=(12,8))
sns.barplot(data=df1,x=df1['date'].dt.day.value_counts(sort=True).ke
plt.title('Barplot of Distribution of last contact day of the month'
plt.xlabel('Last contact day',fontsize=15)
plt.ylabel('No of Clients',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



In [53]:
```
df1['date'].dt.day.describe()
```

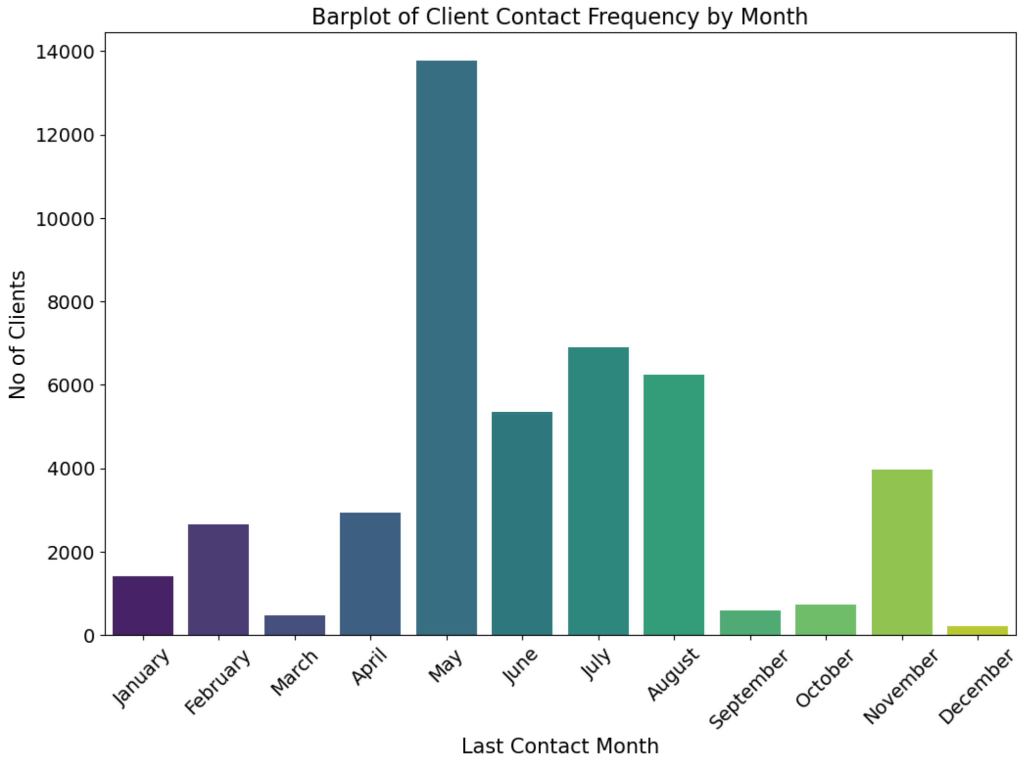
Out[53]:
```
count    45211.000000
mean        15.806419
std          8.322476
min          1.000000
25%          8.000000
50%         16.000000
75%         21.000000
max         31.000000
Name: date, dtype: float64
```

Conclusion:

1. The distribution of last contact days is not uniform across the month.
2. There is a significant peak around the middle of the month, specifically on day 20, indicating a higher frequency of client contacts on that day.
3. The beginning and the end of the month show lower frequencies of contact.
4. Notably, the 31st has the lowest frequency, which could be due to fewer months having this date.
5. Days 1 and 10 also exhibit lower activity compared to their neighboring days.

11. Variation of last contact month among clients

In [54]:
```
order = ['January','February','March','April','May','June','July','A
plt.figure(figsize=(12,8))
sns.barplot(data=df1,x=df1['date'].dt.month_name().value_counts().ke
plt.title('Barplot of Client Contact Frequency by Month',fontsize=16
plt.xlabel('Last Contact Month',fontsize=15)
plt.ylabel('No of Clients',fontsize=15)
plt.xticks(rotation = 45,fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



Conclusions:

1. The contact frequency is significantly higher in May than in any other month, suggesting that this is a peak period for the marketing campaign.
2. The lowest contact frequencies are observed in the months of January, February, and December, indicating a possible seasonal downturn in marketing activities.

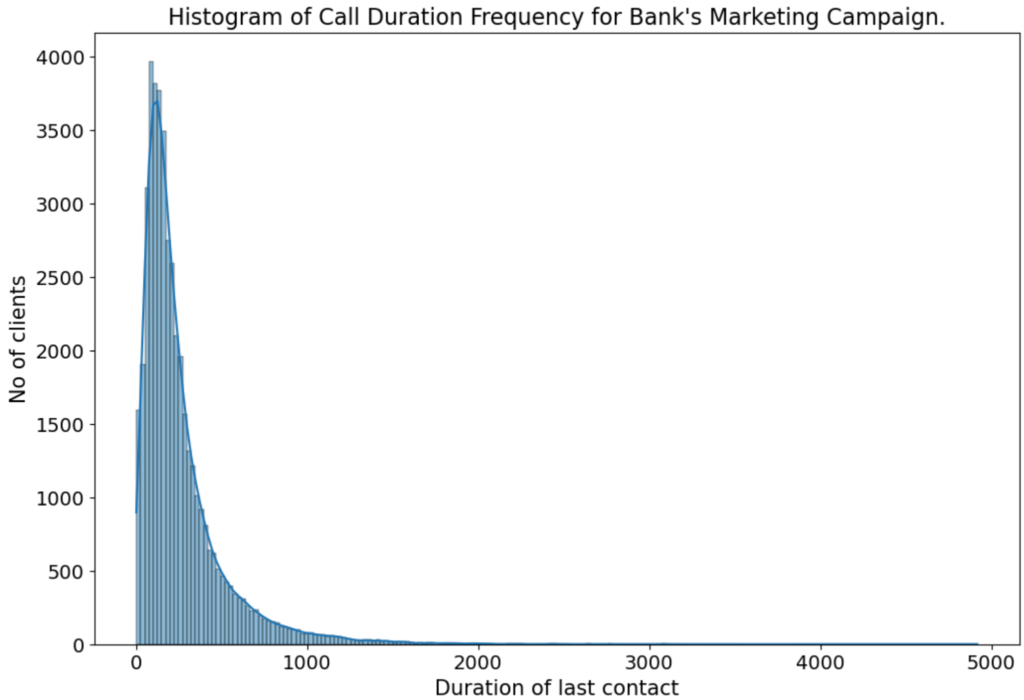
3. The months of June, July, August, and November show a moderate level of contact frequency.
4. There's a notable drop in contact frequency after May, with the numbers gradually increasing again towards August, followed by a decrease towards the end of the year.

12. Distribution of duration of last contact

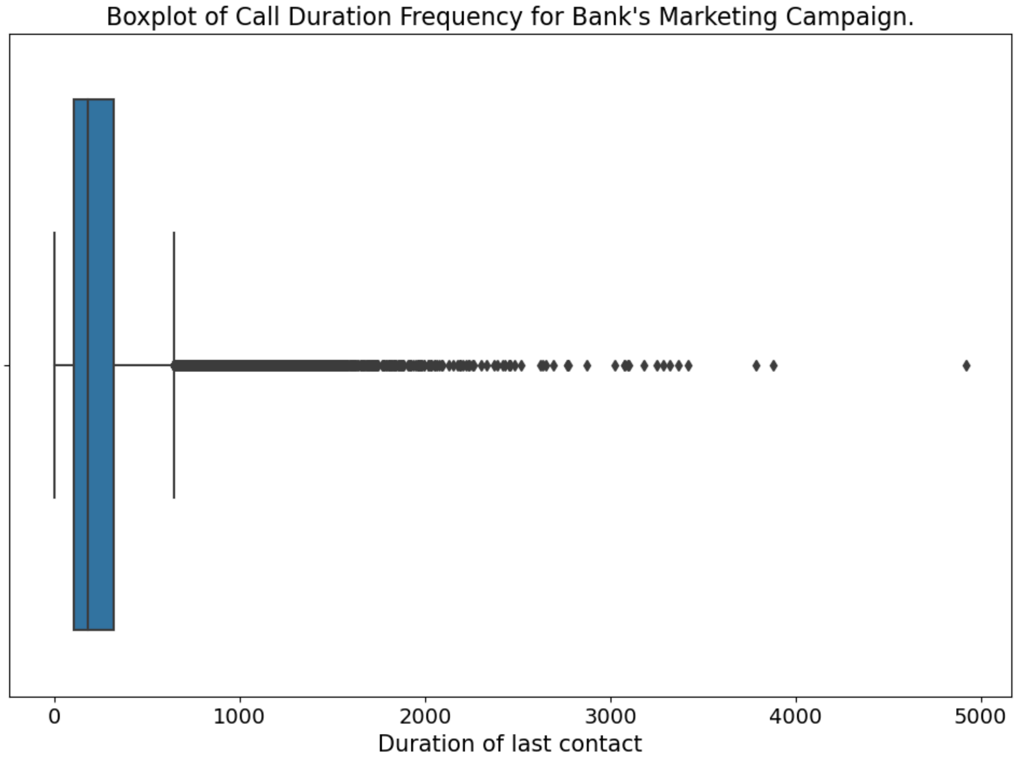
In [55]:
```
df1['duration'].describe()
```

Out[55]:
```
count    45211.000000
mean       258.163080
std        257.527812
min          0.000000
25%        103.000000
50%        180.000000
75%        319.000000
max       4918.000000
Name: duration, dtype: float64
```

In [56]:
```
plt.figure(figsize=(12,8))
sns.histplot(df1['duration'],bins=200,kde=True)
plt.title("Histogram of Call Duration Frequency for Bank's Marketing
plt.xlabel('Duration of last contact',fontsize=15)
plt.ylabel('No of clients',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



In [57]:
```
plt.figure(figsize=(12,8))
sns.boxplot(data=df1,x='duration')
plt.title("Boxplot of Call Duration Frequency for Bank's Marketing C
plt.xlabel('Duration of last contact',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

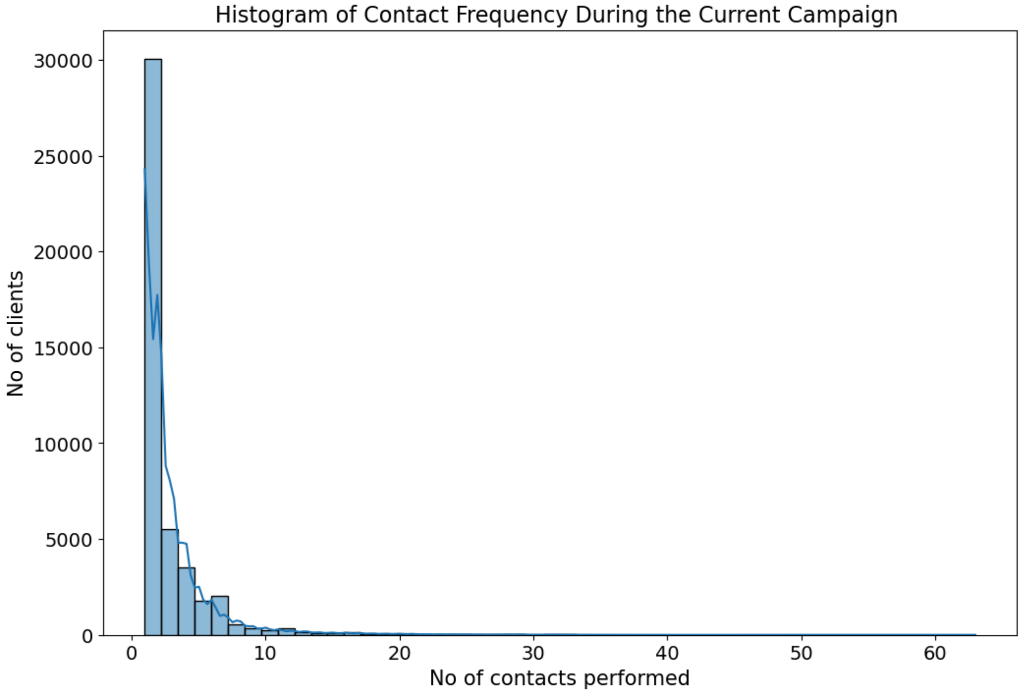


Conclusions:`

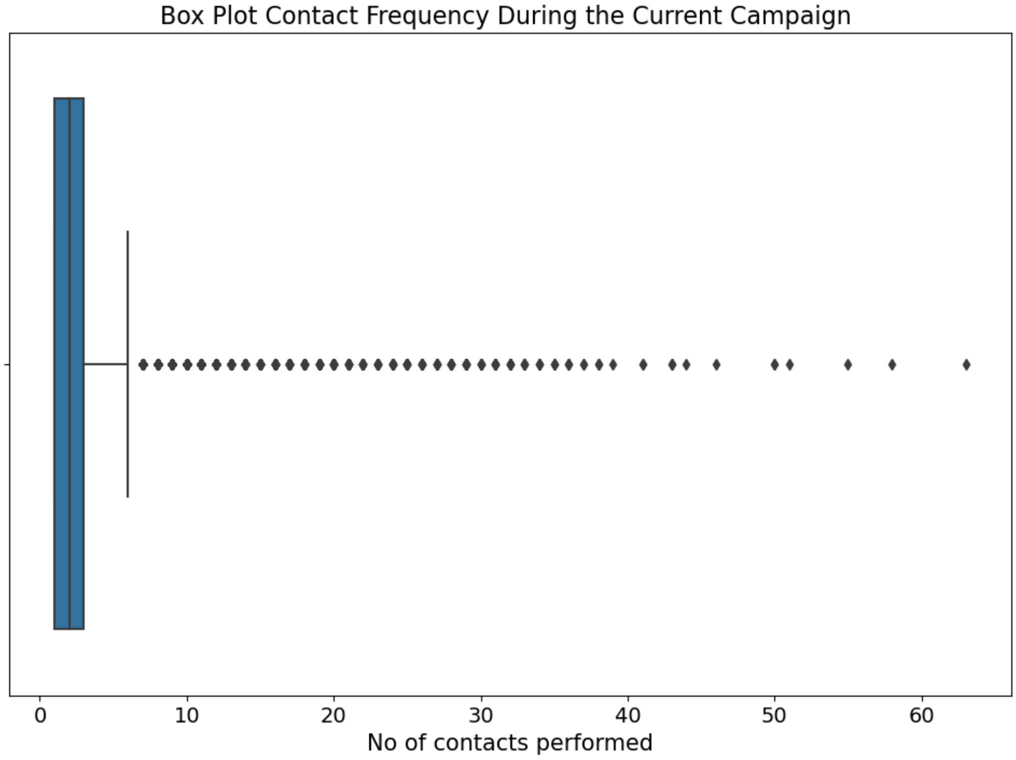
1. The mean call duration is 258s.
2. The distribution is heavily right-skewed, indicating that most calls were relatively short, with a steep decrease in frequency as call duration increases.
3. There is a high frequency of very short calls, with the number of calls declining rapidly as the duration lengthens.
4. Very few calls had a very long duration, which suggests that extended conversations were rare in this campaign.
5. The vast majority of contacts were brief, possibly underlining the efficiency of the call center or a focus on quick interactions.
6. The pattern might indicate that the standard call was meant to be brief, with only specific circumstances leading to longer discussions.

No. of contact performed during the campaign for each client

In [58]:
```
plt.figure(figsize=(12,8))
sns.histplot(df1['campaign'],bins=50,kde=True)
plt.title('Histogram of Contact Frequency During the Current Campaig
plt.xlabel('No of contacts performed',fontsize=15)
plt.ylabel('No of clients',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



In [59]:
```
plt.figure(figsize=(12,8))
sns.boxplot(data=df1,x='campaign')
plt.title('Box Plot Contact Frequency During the Current Campaign ',
plt.xlabel('No of contacts performed',fontsize = 15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

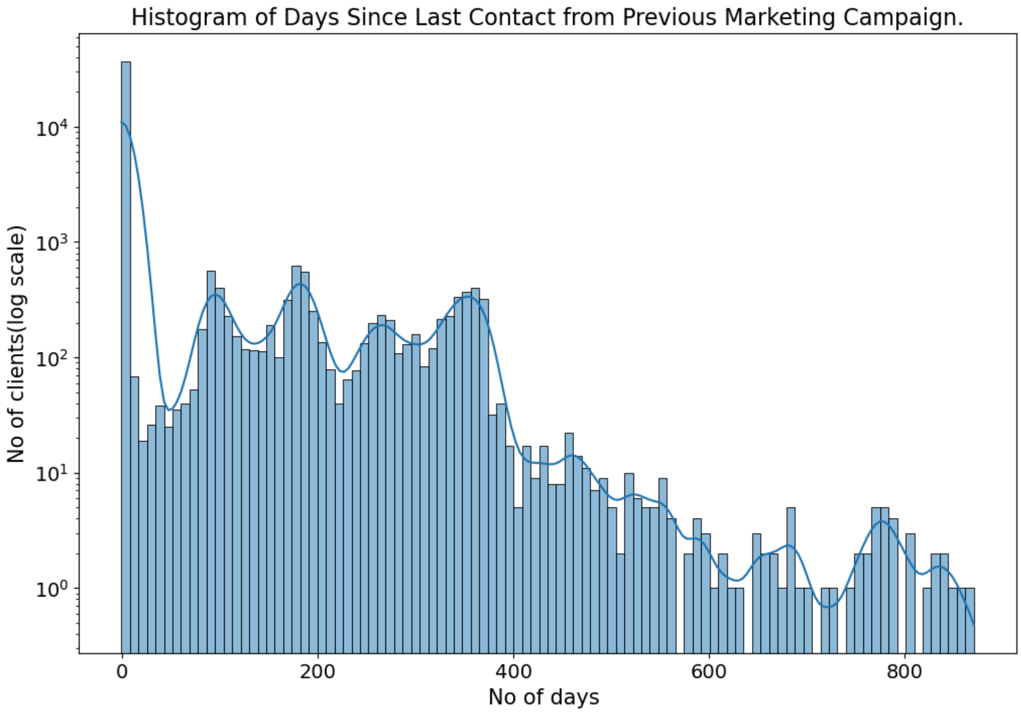


Conclusions:

1. The data is highly positively skewed.
2. The vast majority of clients were contacted a few times, with a sharp decrease in the number of clients as the number of contacts increases.
3. The highest proportion of clients(86.46%) received less than 5 contacts during the campaign.
4. A very small number of clients were contacted more than 20 times, which indicates that such extensive contact is very rare.
5. The distribution of contacts is extremely skewed to the right, suggesting that the campaign strategy primarily focused on a lower number of contacts per client.
6. There is a notable number of outliers where clients were contacted many more times than the median.

14. Distribution of the number of days passed since the client was last contacted from a previous campaign

In [60]:
```
plt.figure(figsize=(12,8))
sns.histplot(df1['pdays'],bins=100,kde=True)
plt.title('Histogram of Days Since Last Contact from Previous Market
plt.xlabel('No of days ',fontsize=15)
plt.ylabel('No of clients(log scale)',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.yscale('log')
plt.show()
```

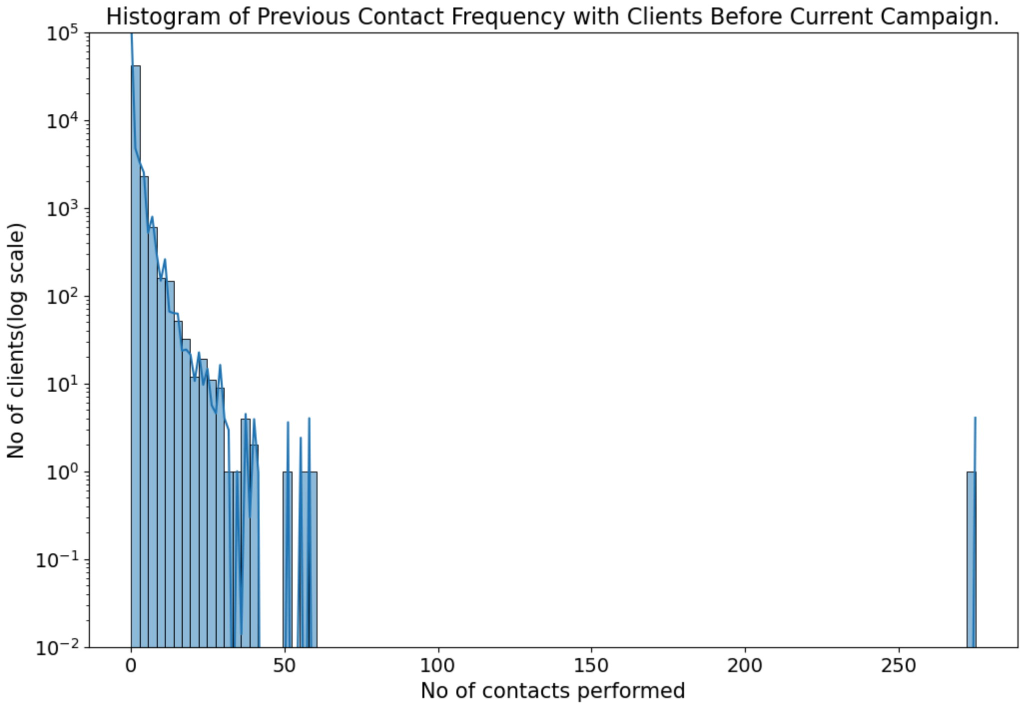


```
Conclusions:
```

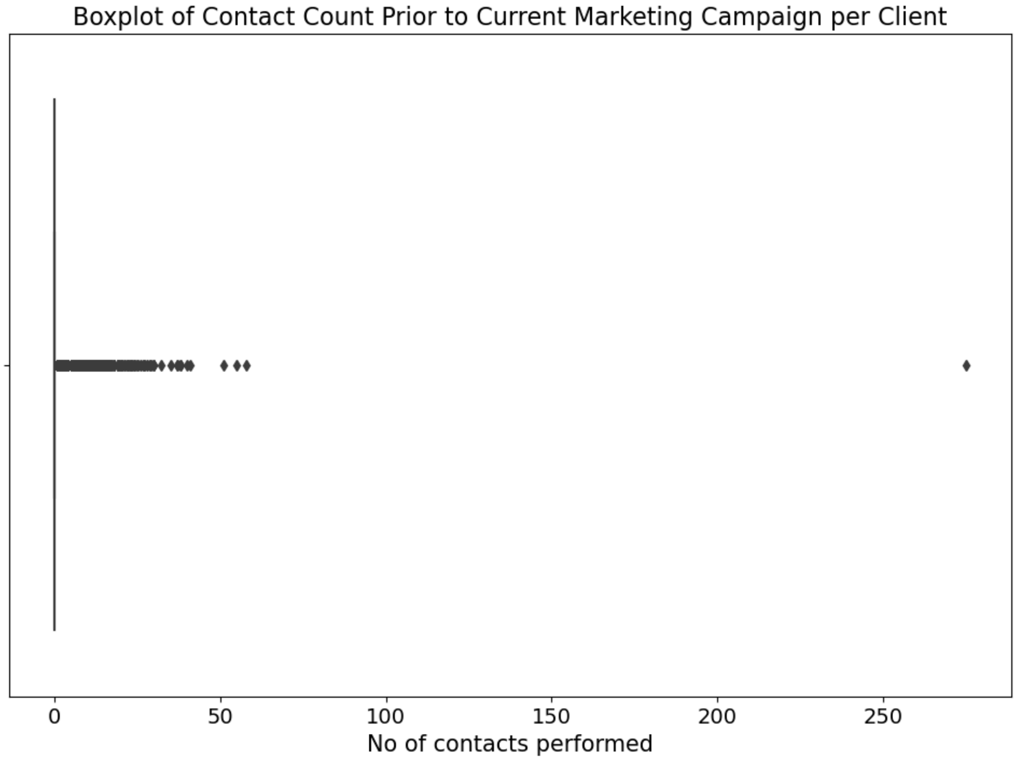
1. Most of the clients(81.73%) have never been contacted before.
2. The data is highly positively skewed.
3. There are relatively few clients who have been contacted after a gap, with the number decreasing sharply as the number of days increases.
4. There is a very long tail to the distribution, indicating that while most recent contacts are quite recent, there are some clients who haven't been contacted for a very long time.
5. The presence of outliers indicates that there are exceptions where clients had not been contacted for a long period before the current campaign.
6. There is a sharp peak at or near zero, indicating that a significant number of clients were contacted recently or not at all since the previous campaign.
7. There are some minor peaks later on, suggesting there might be specific times when re-contacting efforts were concentrated.
8. Overall, the distribution is skewed to the right, reinforcing the idea that most re-contacting efforts occur after a shorter interval or that many clients are new and have not been contacted before the current campaign.

15. No of contacts that were performed before the current campaign for each client

In [61]:
```
plt.figure(figsize=(12,8))
sns.histplot(df1['previous'],bins=100,kde=True)
plt.title('Histogram of Previous Contact Frequency with Clients Befo
plt.xlabel('No of contacts performed',fontsize=15)
plt.ylabel('No of clients(log scale)',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.ylim((0,100))
plt.yscale("log")
plt.ylim(bottom=0.01,top=1e5)
plt.show()
```



In [62]:
```
plt.figure(figsize=(12,8))
sns.boxplot(data=df1,x='previous')
plt.title('Boxplot of Contact Count Prior to Current Marketing Campa
plt.xlabel('No of contacts performed',fontsize=15)
plt.xticks(fontsize=14)
plt.show()
```

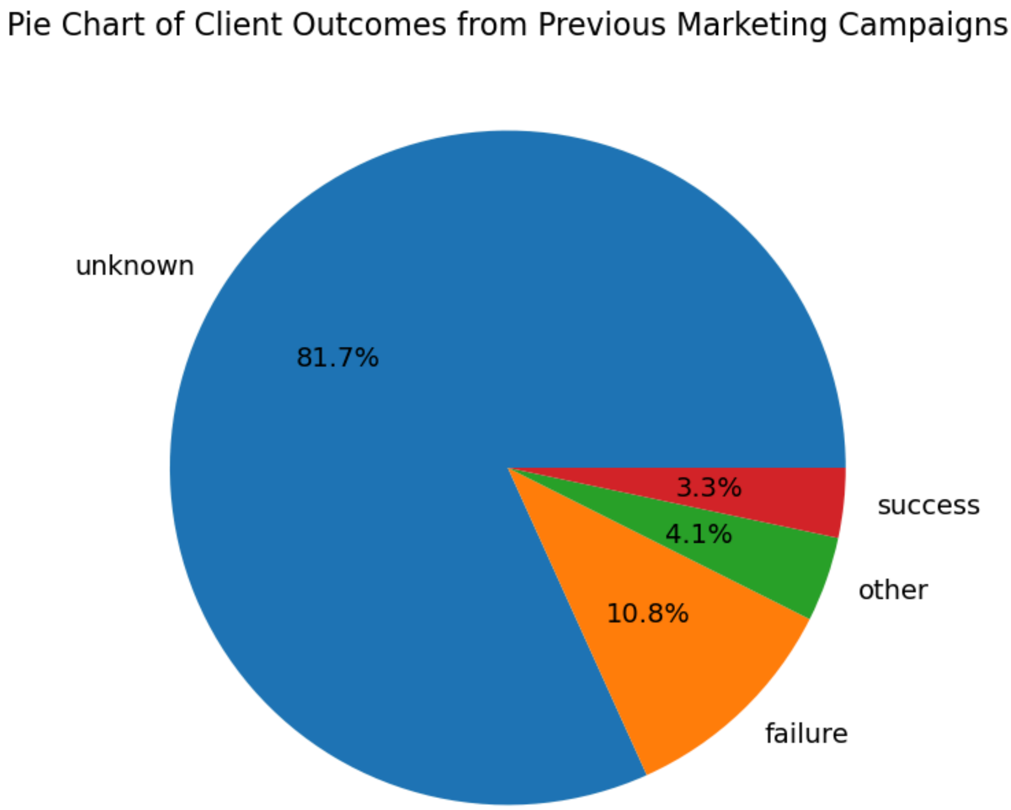


Conclusions:

1. The data is highle positively skewed.
2. The overwhelming majority of clients(81.73%) had zero contacts before the current campaign, suggesting a large number of new engagements or a policy of minimal prior contact.
3. There is a steep drop-off in frequency as the number of previous contacts increases, indicating that repeated outreach to the same clients was relatively uncommon.
4. Very few clients had a high number of contacts, as evidenced by the long tail that extends to the right, which implies that only a select few clients were contacted repeatedly.
5. The distribution is highly right-skewed, meaning that the bank's contact strategy might be focused more on acquiring new clients or those with less prior interaction.
6. Overall, the bank’s outreach strategy likely prioritizes new engagements over repeated contacts with the same clients.
7. There are a significant number of outliers, implying that while most clients had minimal contact, a few had a much higher number of contacts.

16. Outcomes of the previous marketing campaigns

In [63]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['poutcome'].value_counts().tolist(),labels=df1['poutcome
plt.title('Pie Chart of Client Outcomes from Previous Marketing Camp
plt.show()
```

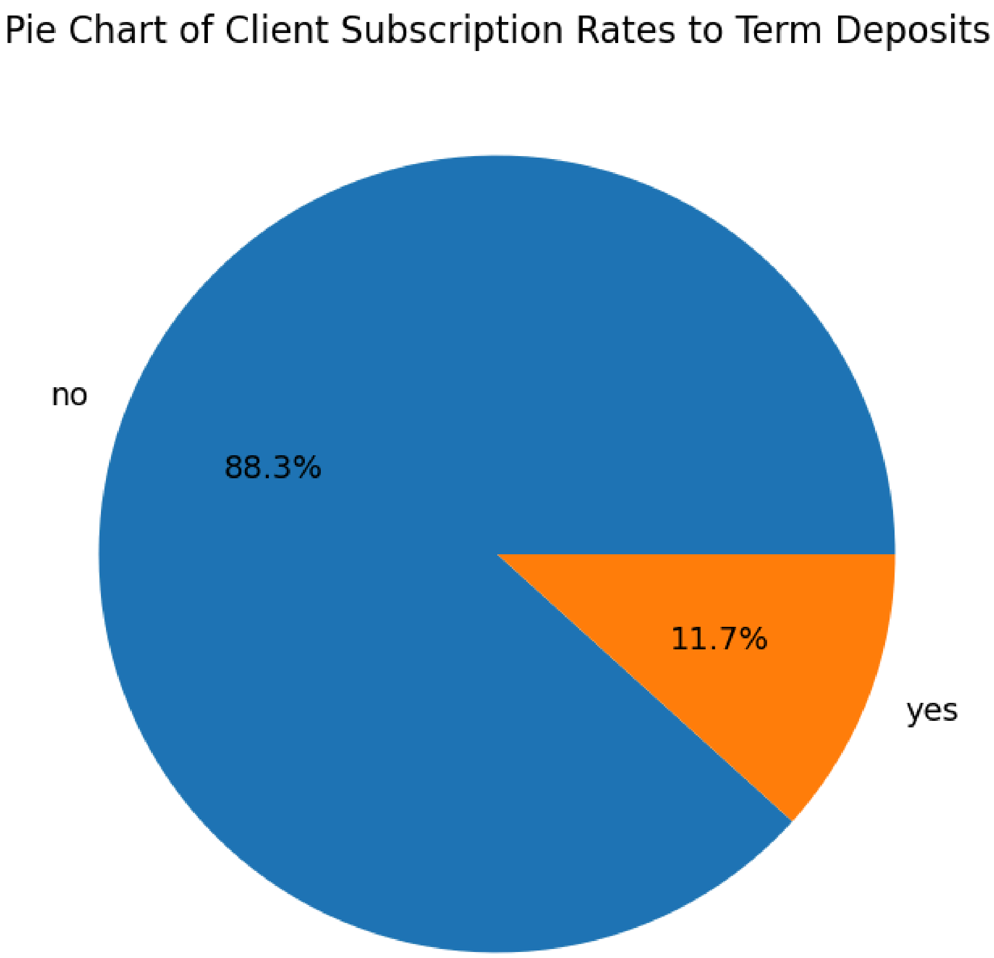


Conclusions:

1. The vast majority of the previous campaign outcomes are unknown, which comprises 81.7% of the total, indicating a lack of data on past client engagement or response.
2. Only a small fraction of clients have a known outcome from previous campaigns, with 10.8% labeled as failures and 3.3% as successes.
3. An even smaller segment, 4.1%, is categorized as other, which might include outcomes that are neither clearly successful nor outright failures.
4. This distribution suggests that there is a significant opportunity for the bank to improve its tracking and analysis of campaign outcomes to better understand client behaviors and patterns.

17. Distribution of clients who subscribed to a term deposit vs. those who did not

In [64]:
```
plt.figure(figsize=(12,8))
plt.pie(df1['y'].value_counts().tolist(),labels=df1['y'].value_count
plt.title('Pie Chart of Client Subscription Rates to Term Deposits',
plt.show()
```




```
Conclusions:
```

1. A significant majority, 88.3%, of clients did not subscribe to a term deposit, indicating a relatively low conversion rate for the campaign.
2. The minority, 11.7%, represents the clients who did subscribe, highlighting the successful conversions.
3. The large disparity between subscribers and non-subscribers suggests room for improvement in targeting or product offering to increase the subscription rate.
4. Strategies to convert the large segment of non-subscribers could include personalized follow-ups, tailored financial products, or incentives.

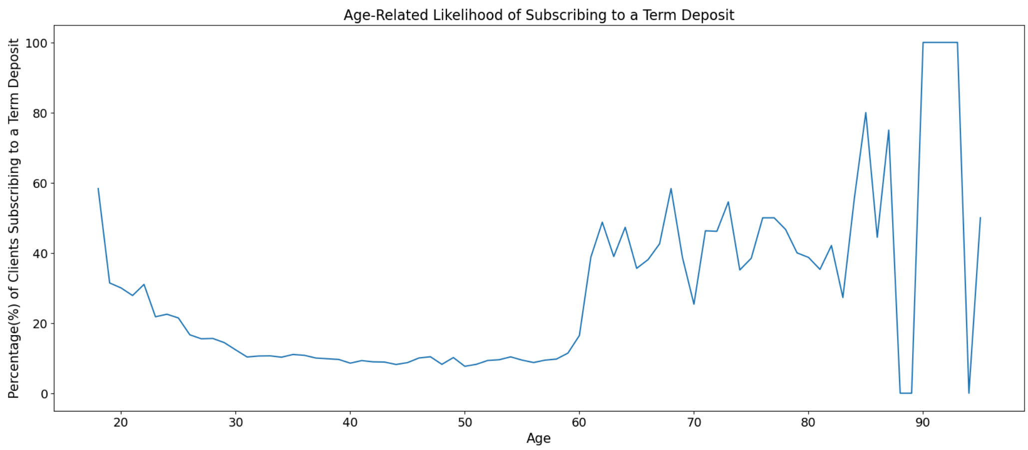
18. Correlations between different attributes and the likelihood of subscribing to a term deposit

a) age vs y

In [65]:
```
df1['age'].value_counts().keys().sort_values()
```

Out[65]:
```
Int64Index([18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 3
1, 32, 33, 34,
            35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 4
8, 49, 50, 51,
            52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 6
5, 66, 67, 68,
            69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 8
2, 83, 84, 85,
            86, 87, 88, 89, 90, 92, 93, 94, 95],
           dtype='int64')
```

In [66]:
```
x = df1.groupby('age')['y'].value_counts().sort_index().tolist()
ages = [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
        36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 5
        54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 6
        72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 8
        90, 92, 93, 94, 95]
y=[]
for i in range (1,78):
    y.append(x[2*i-1])
plt.figure(figsize=(20,8))
plt.title('Age-Related Likelihood of Subscribing to a Term Deposit',
sns.lineplot(y/df1.groupby('age').count()['y']*100)
plt.xlabel('Age',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```




```
Conclusions:
```

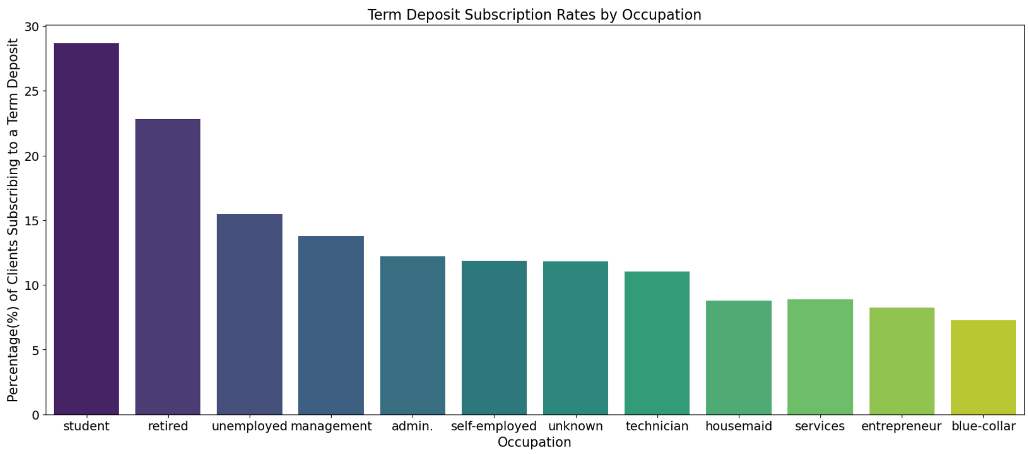
- Younger clients, particularly those in the 18 to 30 age range, show a lower likelihood of subscribing to a term deposit, which could indicate differing financial priorities or a lack of targeted marketing.
- There is a general increase in subscription rates among clients as age increases, particularly noticeable in clients aged 60 and above.
- The highest percentages of subscription are found in the oldest age brackets, suggesting that term deposits might be more appealing to clients as they approach or are in retirement, possibly due to a greater focus on savings and lower-risk financial products.
- The graph indicates an opportunity to tailor financial advice and product offerings to specific age groups, enhancing the appeal to younger clients while maintaining engagement with older clients.

b) job vs y

In [67]:
```
df1['job'].cat.categories
```

Out[67]:
```
Index(['admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'mana
gement',
       'retired', 'self-employed', 'services', 'student', 'technic
ian',
       'unemployed', 'unknown'],
      dtype='object')
```

In [68]:
```
x = df1.groupby('job')['y'].value_counts().sort_index().tolist()
jobs=['admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'managem
       'retired', 'self-employed', 'services', 'student', 'technicia
       'unemployed', 'unknown']
y=[]
for i in range (1,13):
    y.append(x[2*i-1])
order = ['student','retired','unemployed','management','admin.','sel
plt.figure(figsize=(20,8))
plt.title('Term Deposit Subscription Rates by Occupation',fontsize=1
sns.barplot(data = df1,x=jobs,y=y/df1.groupby('job').count()['y']*10
plt.xlabel('Occupation',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



```
Conclusions:
```

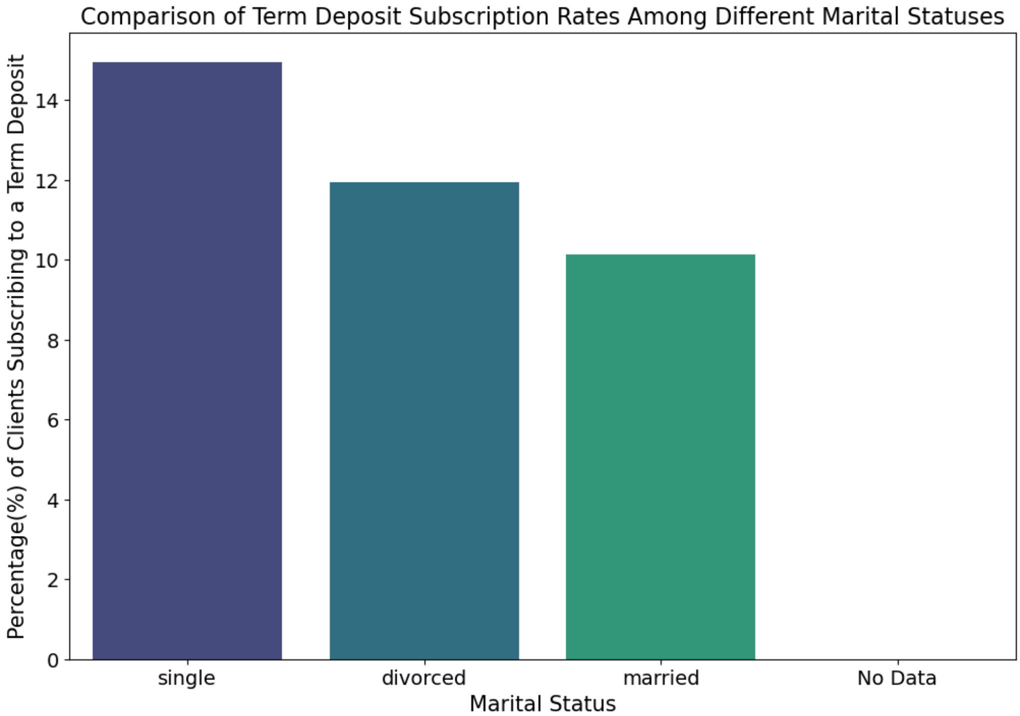
- Retirement seems to significantly increase the likelihood of subscribing to a term deposit, which is likely due to the need for low-risk investments during this life stage.
- Students also show a high likelihood of subscription, possibly indicating good financial awareness or the effect of targeted student banking products.
- Blue-collar workers and entrepreneurs have lower subscription rates, which might suggest a different financial priority or risk preference.
- The 'unknown' category has a moderate subscription rate, indicating a potential area for further data collection to better understand this group.
- Focused financial products and marketing tailored to the needs and financial behaviors of each occupation could improve subscription rates.

c) marital_status vs y

In [69]:
```python
df1['marital_status'].cat.categories
```

Out[69]:
```
Index(['No Data', 'divorced', 'married', 'single'], dtype='objec
t')
```

In [70]:
```
x = df1.groupby('marital_status')['y'].value_counts().sort_index().t
status = ['No Data', 'divorced', 'married', 'single']
y = []
for i in range(1,5):
    y.append(x[2*i-1])
y = y/df1.groupby('marital_status').count()['y']*100
plt.figure(figsize=(12,8))
plt.title('Comparison of Term Deposit Subscription Rates Among Diffe
sns.barplot(data = df1,x=status,y=y,palette='viridis',order = ['sing
plt.xlabel('Marital Status',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



Conclusions:

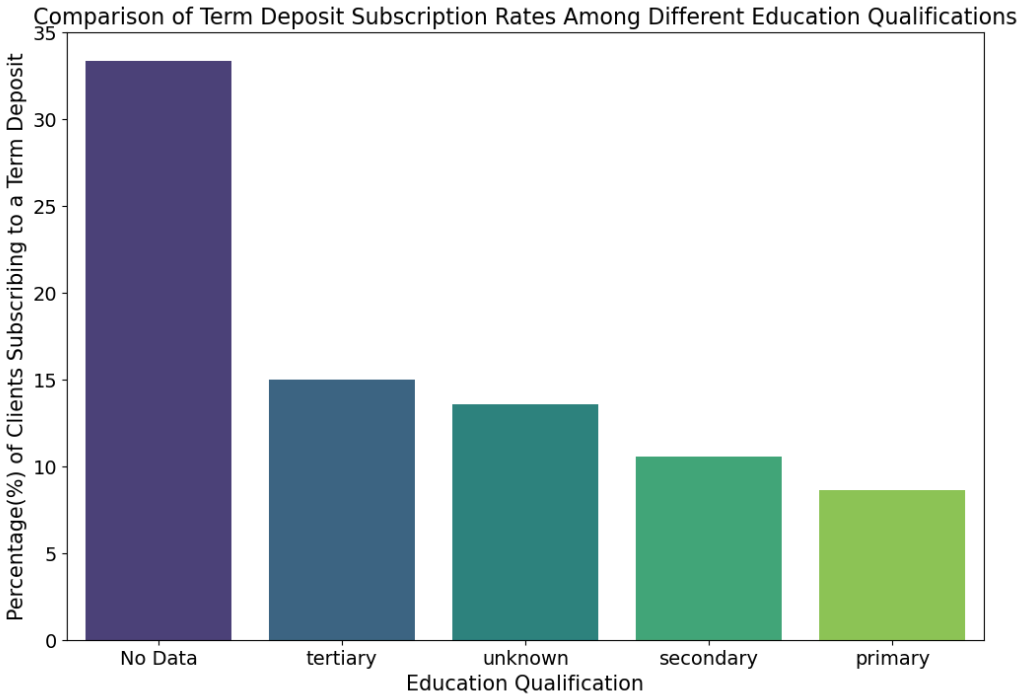
- Single clients have the highest subscription rates to term deposits, suggesting they might have more disposable income or different financial goals compared to other groups.
- Divorced clients show moderately high subscription rates, possibly indicating an increased need for financial security post-divorce.
- Married clients have a lower rate of subscription, which could reflect different financial priorities or obligations such as children and mortgages.
- The "no data" category indicates a gap in the dataset which, if filled, could provide more accurate insights into the correlation between marital status and financial decisions.
- Financial institutions could use these insights to tailor their marketing strategies and product designs to better meet the needs of clients with different marital statuses.

d) education vs y

In [71]:
```
df1['education'].cat.categories
```

Out[71]:
```
Index(['No Data', 'primary', 'secondary', 'tertiary', 'unknown'],
dtype='object')
```

In [72]:
```
x = df1.groupby('education')['y'].value_counts().sort_index().tolist
categories = ['No Data', 'primary', 'secondary', 'tertiary', 'unknow
y = []
for i in range(1,6):
    y.append(x[2*i-1])
y =y/df1.groupby('education').count()['y']*100
plt.figure(figsize=(12,8))
plt.title('Comparison of Term Deposit Subscription Rates Among Diffe
sns.barplot(data = df1,x=categories,y=y,palette='viridis',order = ['
plt.xlabel('Education Qualification',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



Conclusions:

- Clients with tertiary education show a higher likelihood of subscribing to a term deposit, which could reflect better financial literacy or higher income levels that allow for such investments.
- The subscription rate among clients with secondary education is slightly lower than those with tertiary education, suggesting a potential correlation between the level of education and investment decisions.
- Clients with primary education have the lowest subscription rates, possibly indicating a need for more targeted financial education to promote the benefits of term deposits.

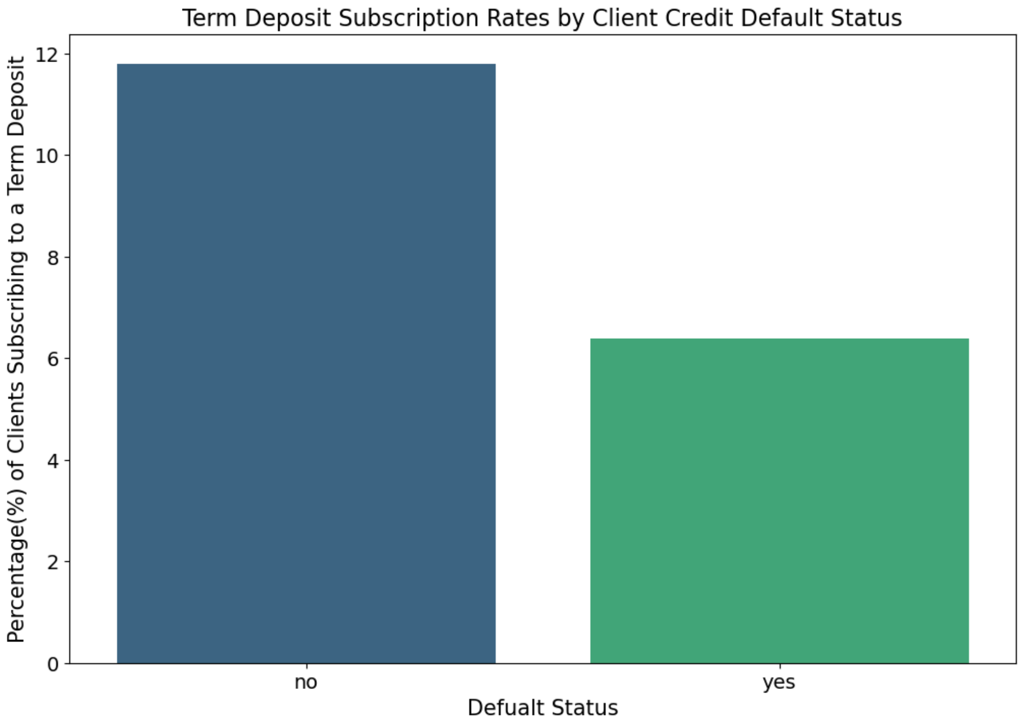
- Some of data is missing or not recorded for clients' education qualifications, which presents a challenge for accurate analysis and targeted marketing strategies.
- Tailored financial advice and products might be more effective if they consider the educational background of the clients, potentially increasing the

'No Data' has the highest bar in this bar chart because the highest percentage of people from it subscribed to the term deposit and not the highest no.

e) default status vs y

In [73]:

```
x = df1.groupby('default')['y'].value_counts().sort_index().tolist()
y = []
y.append(x[1])
y.append(x[3])
plt.figure(figsize=(12,8))
y = y/df1.groupby('default').count()['y']*100
plt.title('Term Deposit Subscription Rates by Client Credit Default
sns.barplot(data = df1,x=['no','yes'],y=y,palette='viridis',order =
plt.xlabel('Defualt Status',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



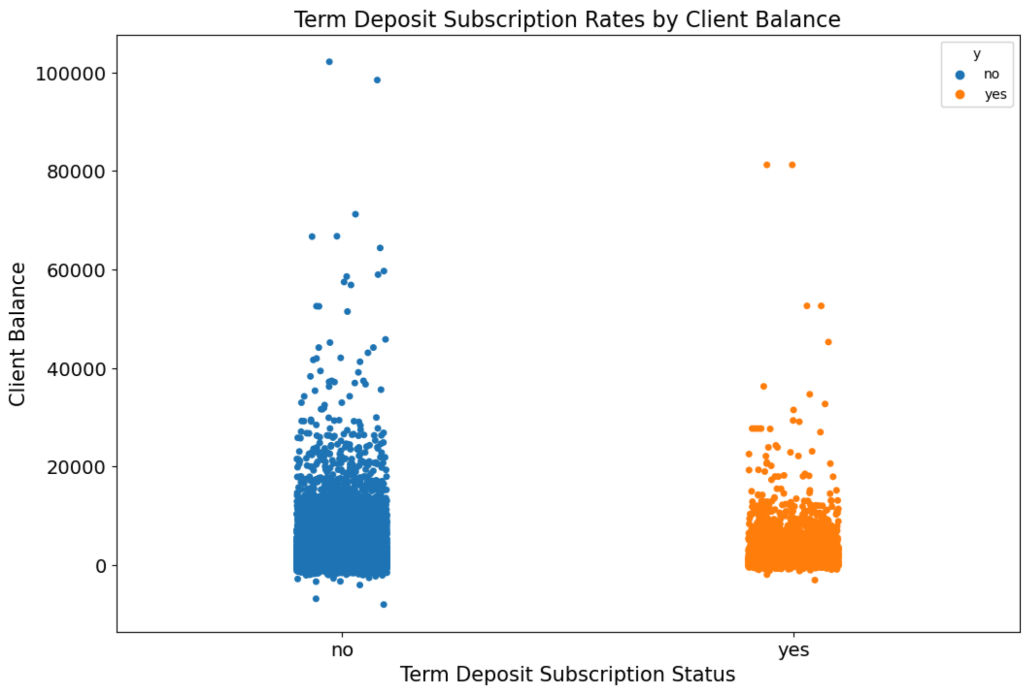

```
Conclusions:
```

- Clients with no default history are significantly more likely to subscribe to a term deposit, which may indicate a general trend of financial responsibility and stability that is attractive to banks for such investments.
- Conversely, clients with a default history show a remarkably lower rate of subscription, suggesting that credit history is a strong indicator of term deposit subscription likelihood.

- The data indicates that default status is a critical factor in the decision-making process for term deposits, and financial institutions may use this as a criterion for marketing such investment products.
- The considerable difference in subscription rates between clients with and without a default history could also inform risk assessment strategies and

f) balance vs y

In [74]:
```
plt.figure(figsize=(12,8))
plt.title('Term Deposit Subscription Rates by Client Balance',fontsi
sns.stripplot(data = df1,x='y',y='balance',hue='y')
plt.xlabel('Term Deposit Subscription Status',fontsize=15)
plt.ylabel('Client Balance',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.xlim((0,20000))
plt.show()
```

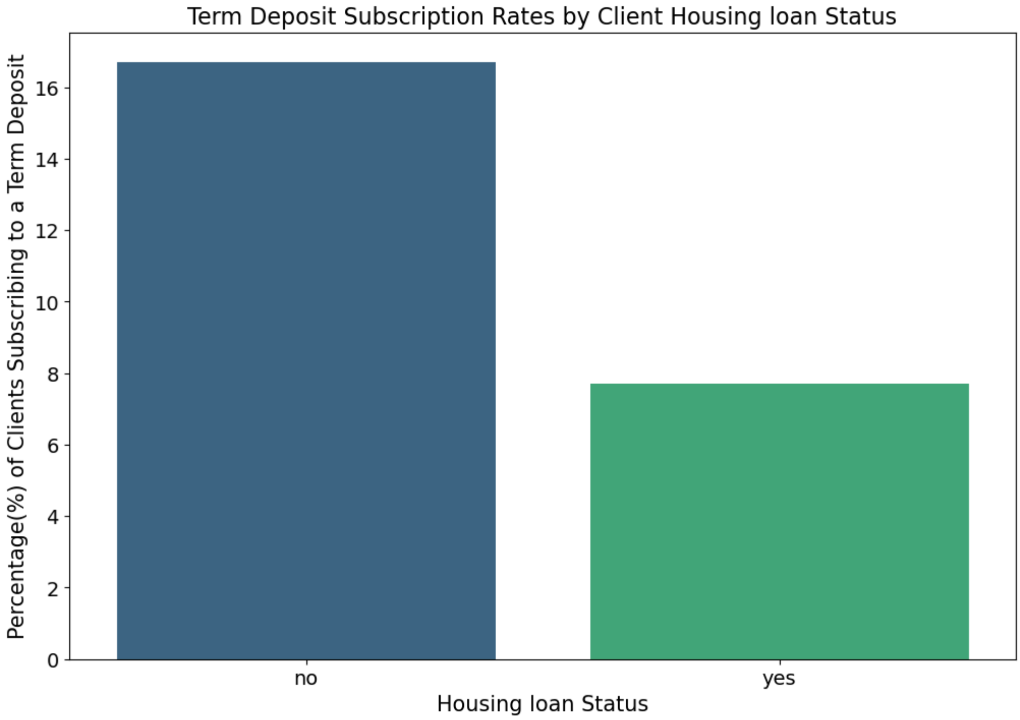



```
Conclusions:
```

- The clients with lower balances(<10000 euros) have an exceptionally low likelihood of subscribing to term deposits.

g) housing vs y

In [75]:
```
x = df1.groupby('housing')['y'].value_counts().sort_index().tolist()
y = []
y.append(x[1])
y.append(x[3])
y = y/df1.groupby('housing').count()['y']*100
plt.figure(figsize=(12,8))
plt.title('Term Deposit Subscription Rates by Client Housing loan St
sns.barplot(data = df1,x=['no','yes'],y=y,palette='viridis',order =
plt.xlabel('Housing loan Status',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

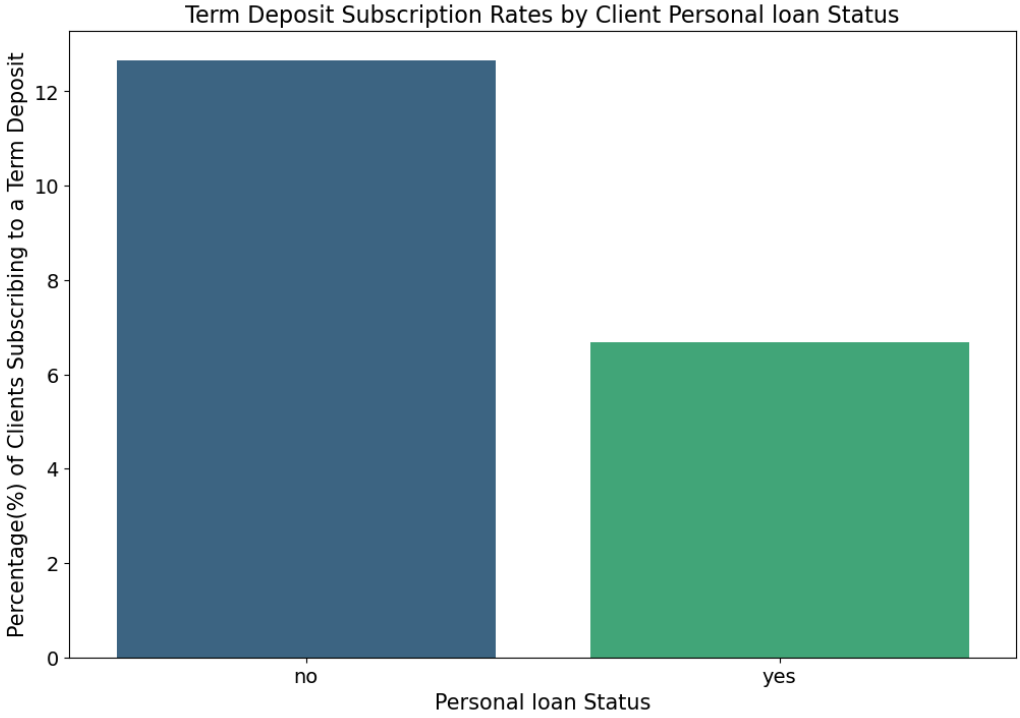


Conclusions:

- Clients without a housing loan appear to have a higher rate of subscribing to a term deposit compared to those with a housing loan.
- The data suggests that financial liabilities such as housing loans may negatively influence a client's decision to commit to a term deposit.
- Financial institutions may consider tailoring their marketing strategies and deposit products for clients based on their loan status.
- A deeper investigation into the reasons why clients with no housing loans are more likely to subscribe could provide insights for product development and customer engagement strategies.
- This chart can serve as a preliminary indication for banks to potentially focus on clients without housing loans for term deposit marketing campaigns.

h) loan vs y

In [76]:
```
x = df1.groupby('loan')['y'].value_counts().sort_index().tolist()
y=[]
y.append(x[1])
y.append(x[3])
y = y/df1.groupby('loan').count()['y']*100
plt.figure(figsize=(12,8))
plt.title('Term Deposit Subscription Rates by Client Personal loan S
sns.barplot(data = df1,x=['no','yes'],y=y,palette='viridis',order =
plt.xlabel('Personal loan Status',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

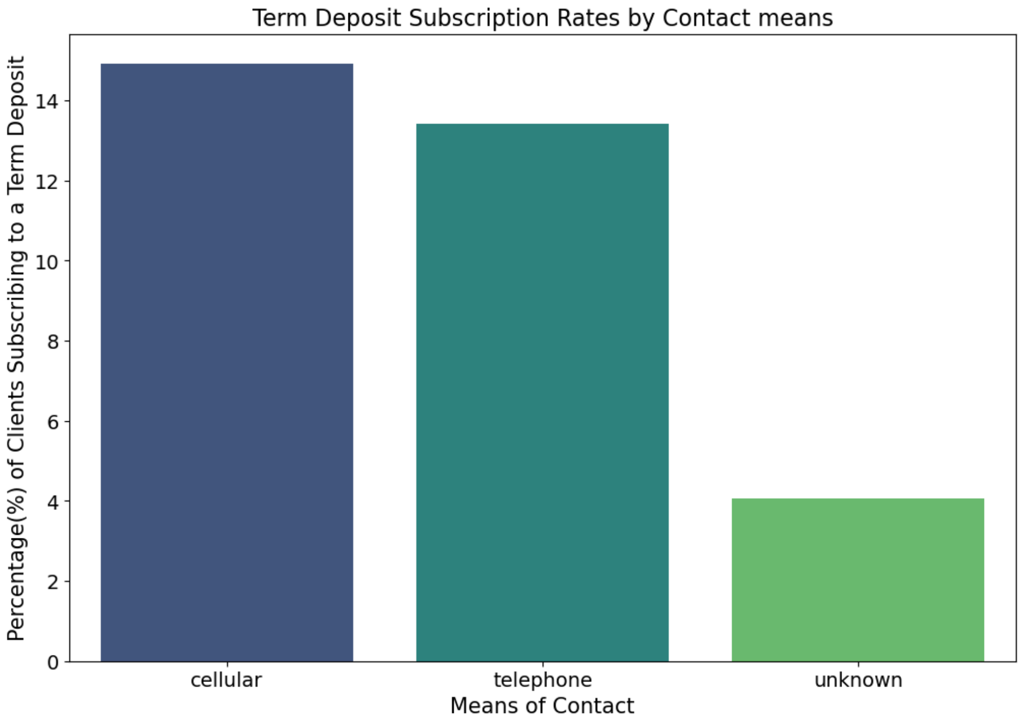


Conclusions:

- A significantly higher percentage of clients without personal loans have subscribed to term deposits compared to those with personal loans.
- The financial burden of a personal loan seems to be inversely related to the likelihood of a client subscribing to a term deposit.
- Clients with no personal loans may have more financial freedom to invest in savings products like term deposits.
- Marketing strategies for term deposits might be more effective if targeted towards clients without personal loan commitments.

i) contact vs y

In [77]:
```
x = df1.groupby('contact')['y'].value_counts().sort_index().tolist()
cat = ['cellular', 'telephone', 'unknown']
y = []
y.append(x[1])
y.append(x[3])
y.append(x[5])
y = y/df1.groupby('contact').count()['y']*100
plt.figure(figsize=(12,8))
plt.title('Term Deposit Subscription Rates by Contact means',fontsiz
sns.barplot(data = df1,x=['cellular','telephone','unknown'],y=y,pale
plt.xlabel('Means of Contact',fontsize=15)
plt.ylabel('Percentage(%) of Clients Subscribing to a Term Deposit',
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

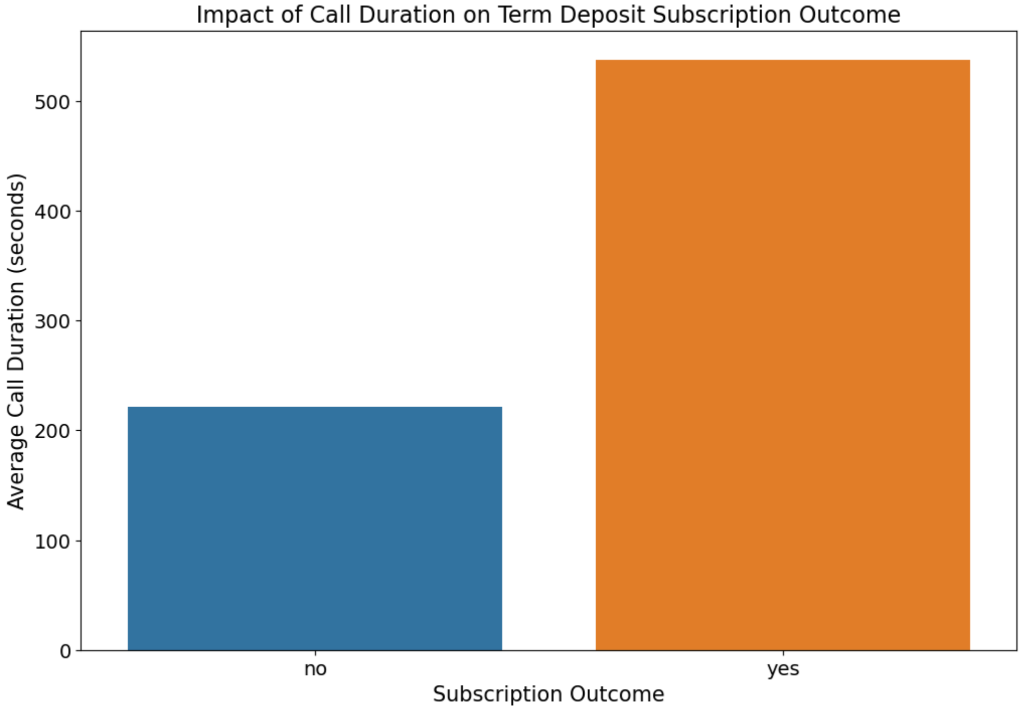



```
Conclusions:
```

- Contact through cellular phones leads to a higher term deposit subscription rate compared to other means of contact.
- The least effective means of contact for term deposit subscriptions is when the means of contact is unknown.
- Telephone contact has a moderate success rate, suggesting that while effective, it may not be as persuasive as cellular contact.
- It may be inferred that personal and direct forms of communication (like cellular phones) could be more effective for marketing term deposits.

j) duration vs y

In [80]:
```
temp_df = df1[['duration','y']]
temp_df = temp_df.groupby('y',observed=True).mean().reset_index()
plt.figure(figsize=(12,8))
sns.barplot(temp_df,x='y',y='duration')
plt.title('Impact of Call Duration on Term Deposit Subscription Outc
plt.ylabel('Average Call Duration (seconds)',fontsize=15)
plt.xlabel('Subscription Outcome',fontsize=15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



Conclusions:

- There's a substantial difference in the average call duration between clients who subscribed to a term deposit and those who did not.
- Longer call durations are associated with a higher likelihood of subscription, which could suggest that more detailed conversations or thorough client engagement correlates with positive outcomes.
- The graph implies that investment in training for customer representatives to effectively engage clients on calls may improve subscription rates.
- It might be beneficial to analyze the content and quality of the calls to understand what aspects contribute to successful conversions.
- This insight can help to refine communication strategies and prioritize call duration as a key performance indicator for sales teams.