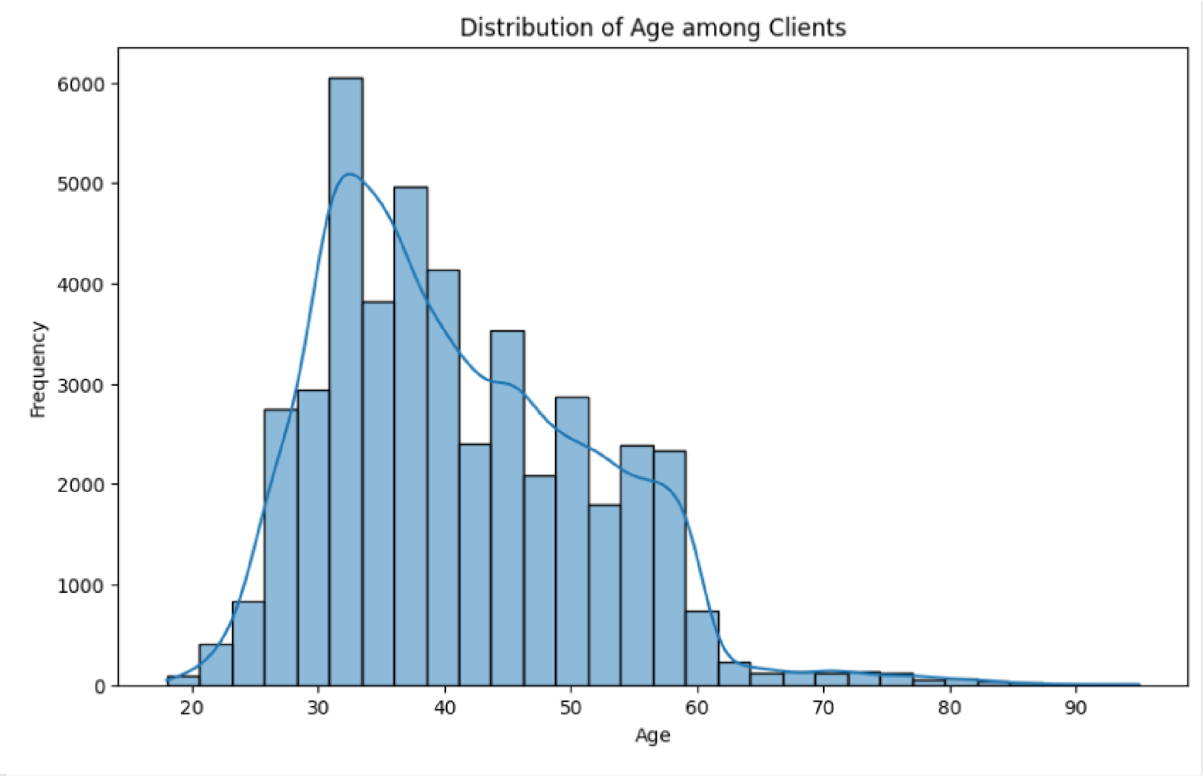
Case Project – Banking

Q.1 What is the distribution of age among the clients?

Sol.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting the age distribution
plt.figure(figsize=(10, 6))
sns.histplot(data['age'], bins=30, kde=True)
plt.title('Distribution of Age among Clients')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



```
# Summary statistics of the age column
print(data['age'].describe())


# Checking for any obvious outliers
print(data['age'].unique())
```

```
count    45216.000000
mean        40.938186
std         10.621249
min         18.000000
25%         33.000000
50%         39.000000
75%         48.000000
max         95.000000
Name: age, dtype: float64
[58 44 33 47 35 28 42 43 41 29 53 57 51 45 60 56 32 25 40 39 52 46 36
49
 59 37 50 54 55 48 24 38 31 30 27 34 23 26 61 22 21 20 66 62 83 75 67
70
 65 68 64 69 72 71 19 76 85 63 90 82 73 74 78 80 94 79 77 86 95 81 18
89
 84 87 92 93 88]
```

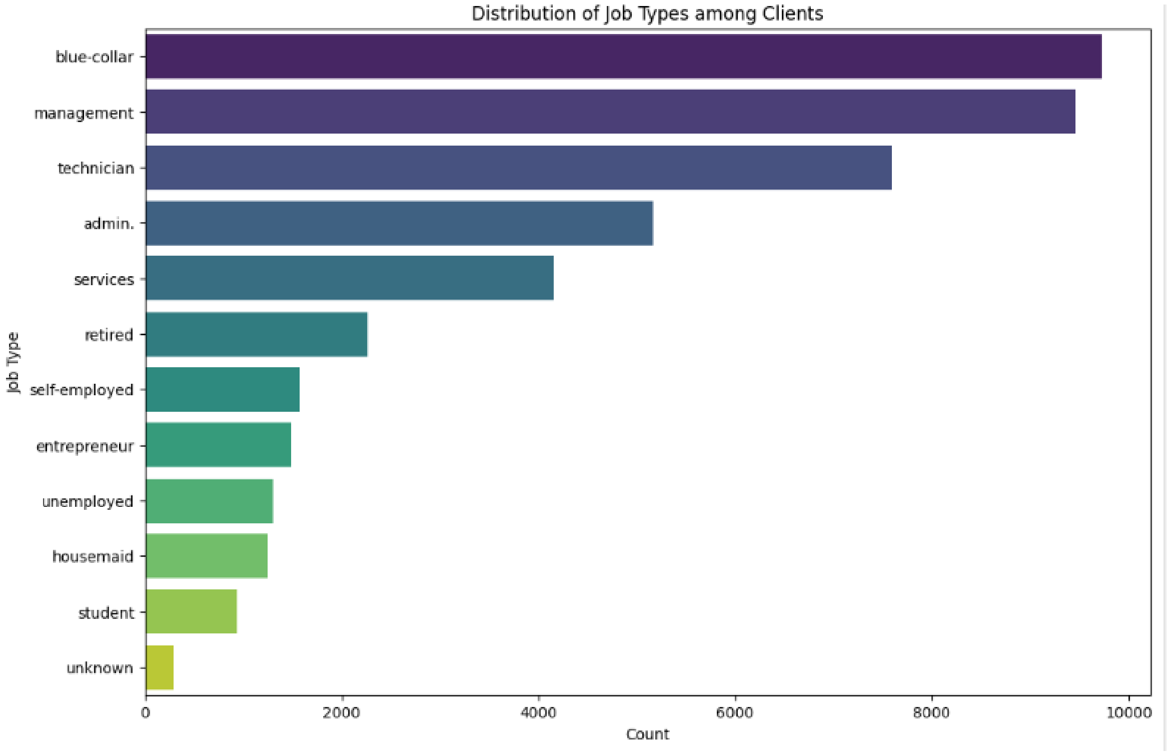
Q.2 How does the job type vary among the clients?

```
# Summary of job type column
print(data['job'].value_counts())


# Display the unique job types
print(data['job'].unique())
```

```
job
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: count, dtype: int64
['management' 'technician' 'entrepreneur' 'blue-collar' 'unknown'
 'retired' 'admin.' 'services' 'self-employed' 'unemployed' 'housemaid'
 'student']
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting the job type distribution
plt.figure(figsize=(12, 8))
sns.countplot(y=data['job'], order=data['job'].value_counts().index, palette='viridis')
plt.title('Distribution of Job Types among Clients')
plt.xlabel('Count')
plt.ylabel('Job Type')
plt.show()
```

```
ipython-input-9-43d4d336c813>:6: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.


  sns.countplot(y=data['job'], order=data['job'].value_counts().index,
palette='viridis')
```



Q.3 What is the marital status distribution of the clients?

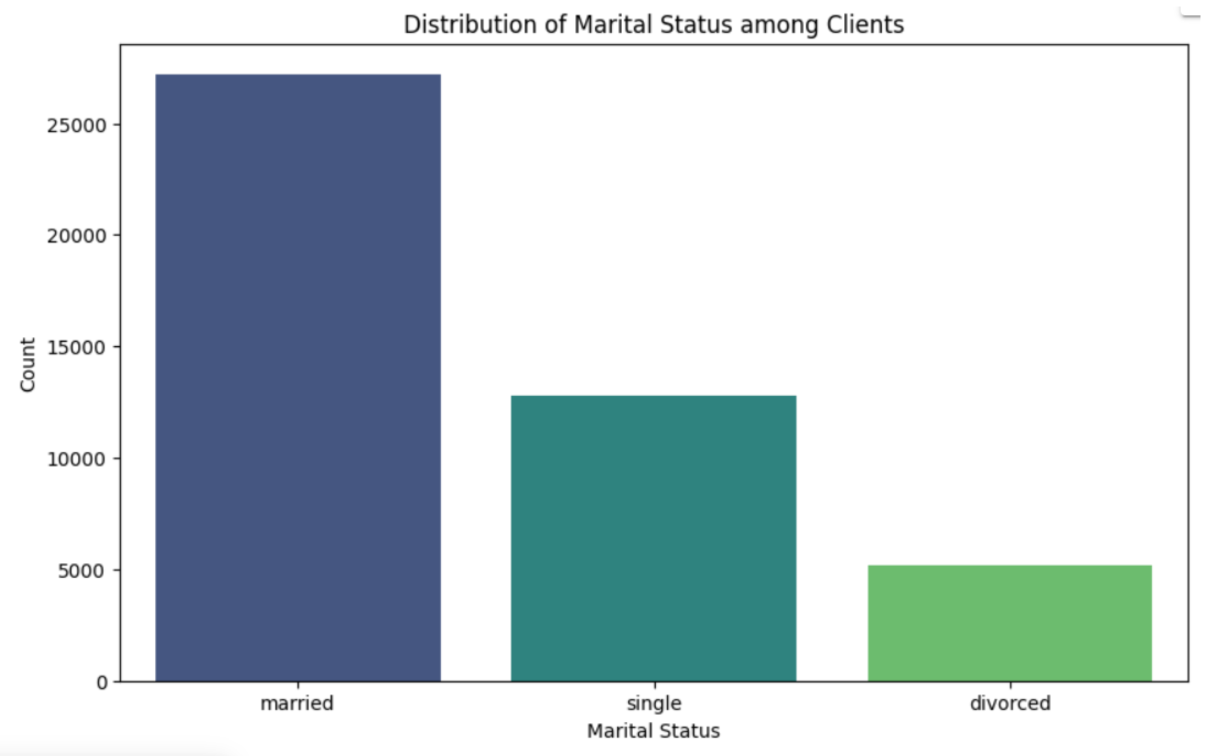
```
# Summary of marital status column
marital_status_counts = data['marital'].value_counts()
print(marital_status_counts)

# Display the unique marital statuses
print(data['marital'].unique())
```

```
marital
married     27216
single      12790
divorced     5207
Name: count, dtype: int64
['married' 'single' 'divorced' nan]
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting the marital status distribution
plt.figure(figsize=(10, 6))
sns.countplot(x=data['marital'], order=data['marital'].value_counts().index,
palette='viridis')
plt.title('Distribution of Marital Status among Clients')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-11-80e6f4bd2829>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
  sns.countplot(x=data['marital'], order=data['marital'].value_counts().index,
palette='viridis')
```



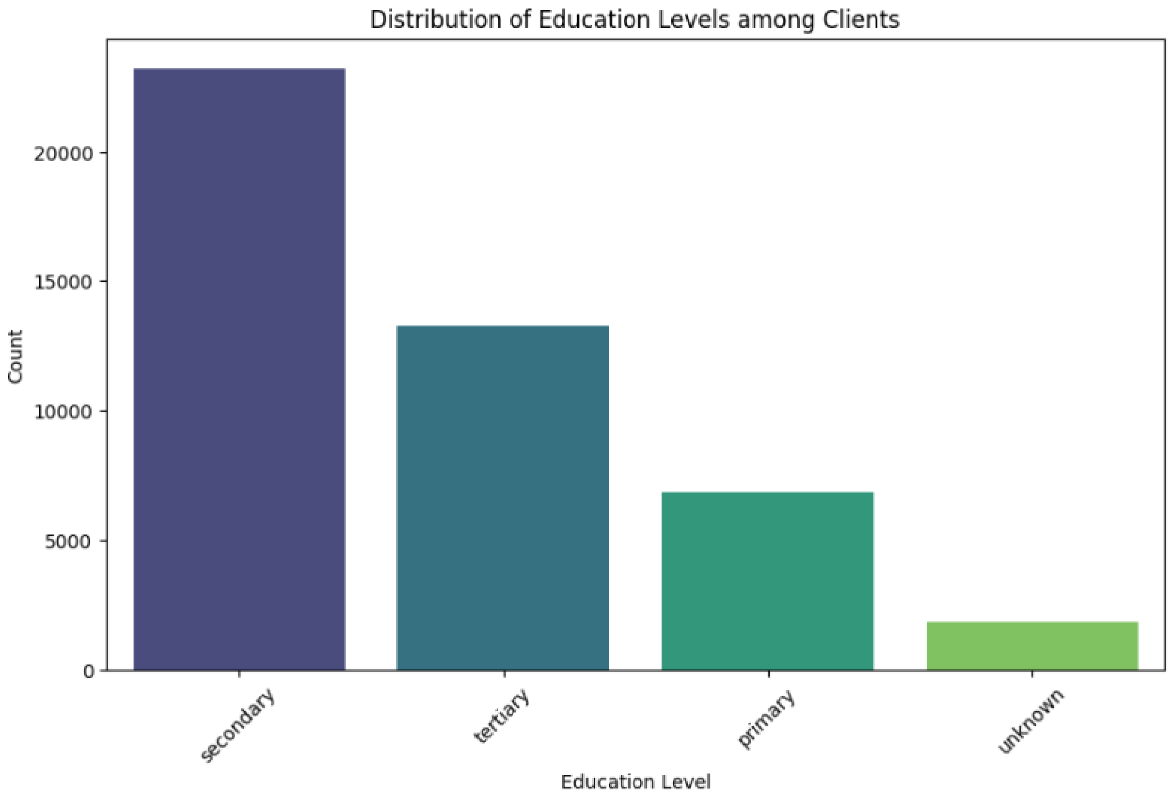
Q. 4 What is the level of education among the clients?

```
# Summary of education column

education_counts = data['education'].value_counts()

print(education_counts)


# Display the unique education levels

print(data['education'].unique())
```

```
education

secondary    23204

tertiary     13301

primary       6851

unknown       1857

Name: count, dtype: int64

['tertiary' 'secondary' 'unknown' 'primary' nan]
```

```
import matplotlib.pyplot as plt

import seaborn as sns


# Plotting the education level distribution

plt.figure(figsize=(10, 6))

sns.countplot(x=data['education'], order=data['education'].value_counts().index, palette='viridis')

plt.title('Distribution of Education Levels among Clients')

plt.xlabel('Education Level')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.


  sns.countplot(x=data['education'], order=data['education'].value_counts().index, palette='viridis')
```



Q.5 What proportion of clients have credit in default?

```
# Summary of default column
default_counts = data['default'].value_counts()
print(default_counts)

# Display the unique values in the default column
print(data['default'].unique())
```

```
default
no     44401
yes      815
Name: count, dtype: int64
['no' 'yes']
```

```
# Calculate the proportion of clients with credit in default
total_clients = len(data)
clients_with_default = default_counts['yes']
proportion_with_default = clients_with_default / total_clients

print(f"Proportion of clients with credit in default: {proportion_with_default:.2%}")
```

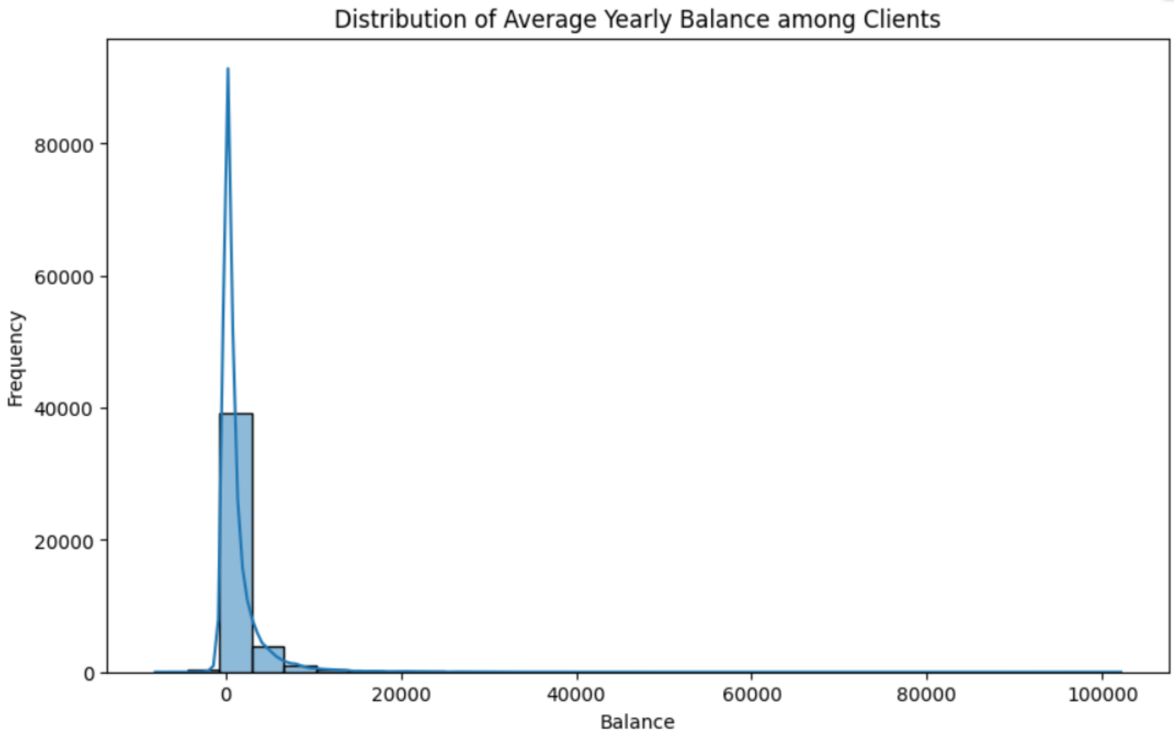
```
Proportion of clients with credit in default: 1.80%
```

Q.6 What is the distribution of average yearly balance among the clients?

```
# Summary statistics of the balance column
balance_summary = data['balance'].describe()
print(balance_summary)

# Checking for any obvious outliers
print(data['balance'].unique())
```

```
count     45216.000000
mean       1362.277844
std        3044.609674
min       -8019.000000
25%          72.000000
50%         448.500000
75%        1428.000000
max      102127.000000
Name: balance, dtype: float64
[ 2143    29     2 ...  8205 14204 16353]
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting the balance distribution
plt.figure(figsize=(10, 6))
sns.histplot(data['balance'], bins=30, kde=True)
plt.title('Distribution of Average Yearly Balance among Clients')
plt.xlabel('Balance')
plt.ylabel('Frequency')
plt.show()
```



Q.7 How many clients have housing loans?

```
#How many clients have housing loans?
# Count of clients with housing loans
housing_loan_counts = data['housing'].value_counts()
print(housing_loan_counts)

# Number of clients with housing loans
num_housing_loans = housing_loan_counts['yes']
print(f"Number of clients with housing loans: {num_housing_loans}")
```

```
housing
yes    25130
no     20086
Name: count, dtype: int64
Number of clients with housing loans: 25130
```

Q.8 How many clients have personal loans?

```
#How many clients have personal loans?
# Count of clients with personal loans
```

```
personal_loan_counts = data['loan'].value_counts()
print(personal_loan_counts)

# Number of clients with personal loans
num_personal_loans = personal_loan_counts['yes']
print(f"Number of clients with personal loans: {num_personal_loans}")
```

```
loan
no     37972
yes     7244
Name: count, dtype: int64
Number of clients with personal loans: 7244
```

Q.9 What are the communication types used for contacting clients during the campaign?

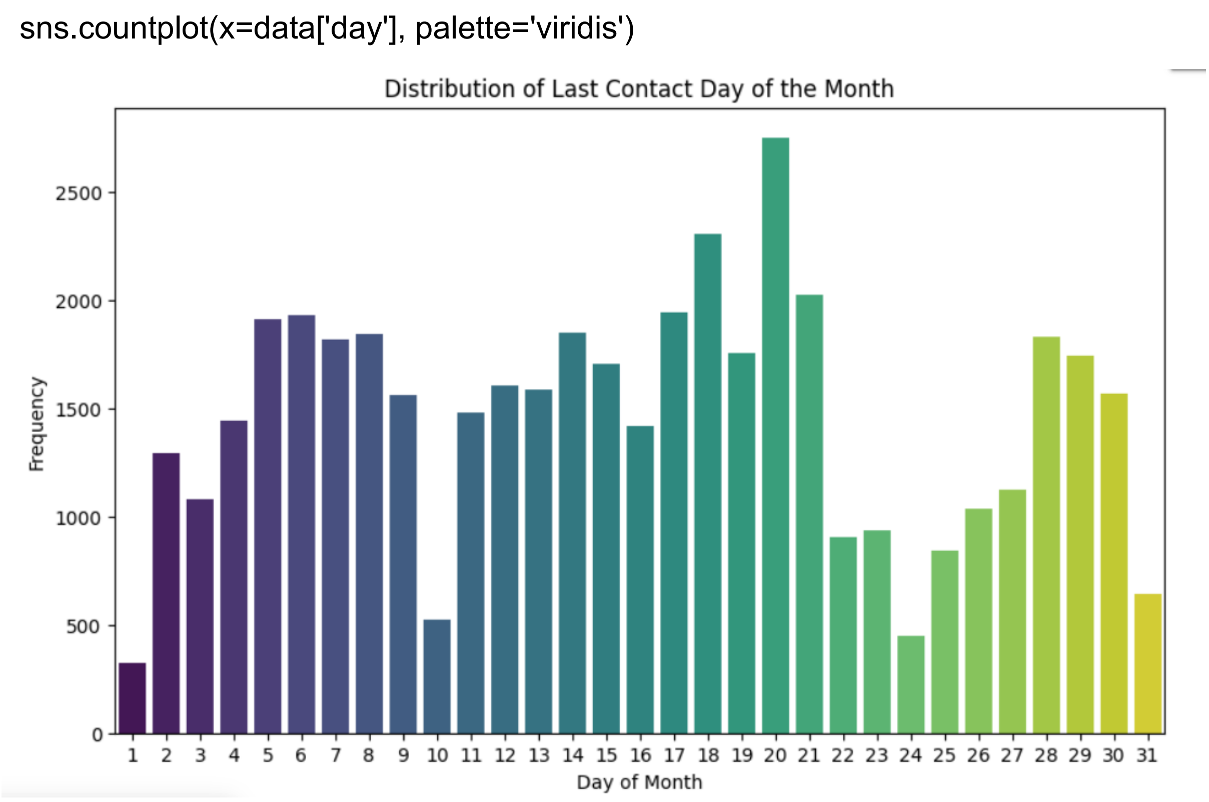
```
#What are the communication types used for contacting clients during the campaign?
# Unique communication types
communication_types = data['contact'].unique()
print(communication_types)

# Count of each communication type
communication_type_counts = data['contact'].value_counts()
print(communication_type_counts)
```

```
['unknown' 'cellular' 'telephone']
contact
cellular     29290
unknown      13020
telephone     2906
Name: count, dtype: int64
```

Q.10 What is the distribution of the last contact day of the month?

```
# What is the distribution of the last contact day of the month?
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting the last contact day of the month distribution
plt.figure(figsize=(10, 6))
sns.countplot(x=data['day'], palette='viridis')
plt.title('Distribution of Last Contact Day of the Month')
plt.xlabel('Day of Month')
plt.ylabel('Frequency')
plt.show()
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```



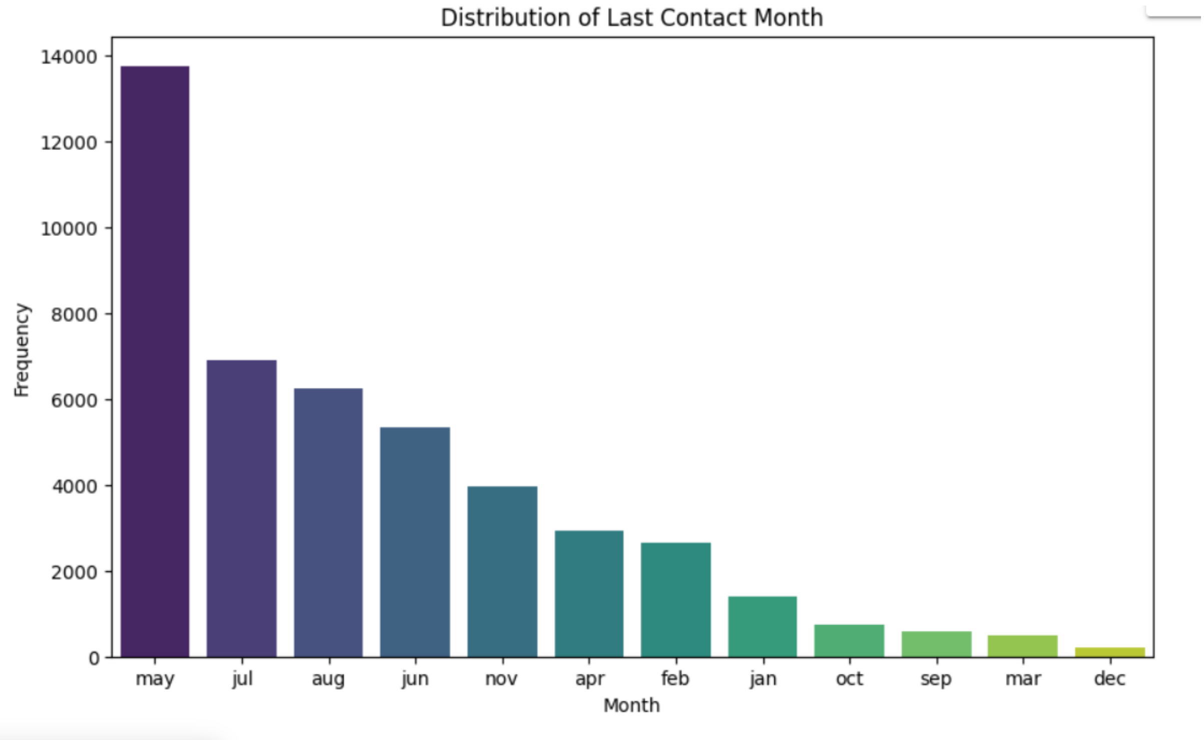
```
  sns.countplot(x=data['day'], palette='viridis')
```

Q.11How does the last contact month vary among the clients?

```
# How does the last contact month vary among the clients?
# Unique contact months
contact_months = data['month'].unique()
print(contact_months)

# Count of each contact month
contact_month_counts = data['month'].value_counts()
print(contact_month_counts)

# Plotting the last contact month distribution
plt.figure(figsize=(10, 6))
sns.countplot(x=data['month'], order=contact_month_counts.index, palette='viridis')
plt.title('Distribution of Last Contact Month')
plt.xlabel('Month')
plt.ylabel('Frequency')
plt.show()
```

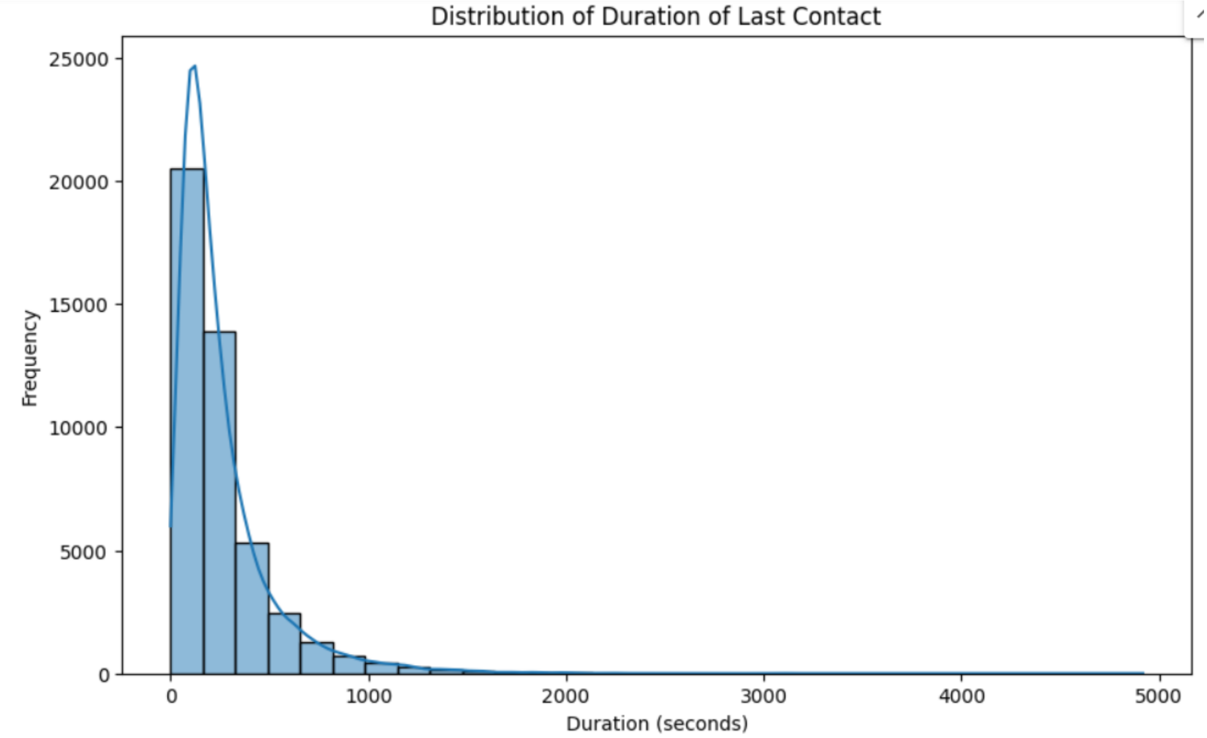
```
['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'jan' 'feb' 'mar' 'apr' 'sep']
month
may    13766
jul     6895
aug     6247
jun     5341
nov     3975
apr     2932
feb     2649
jan     1403
oct      738
sep      579
mar      477
```

```
dec      214
Name: count, dtype: int64
<ipython-input-7-98ba277aad48>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=data['month'], order=contact_month_counts.index, palette='viridis')
```



Q.12 What is the distribution of the duration of the last contact?

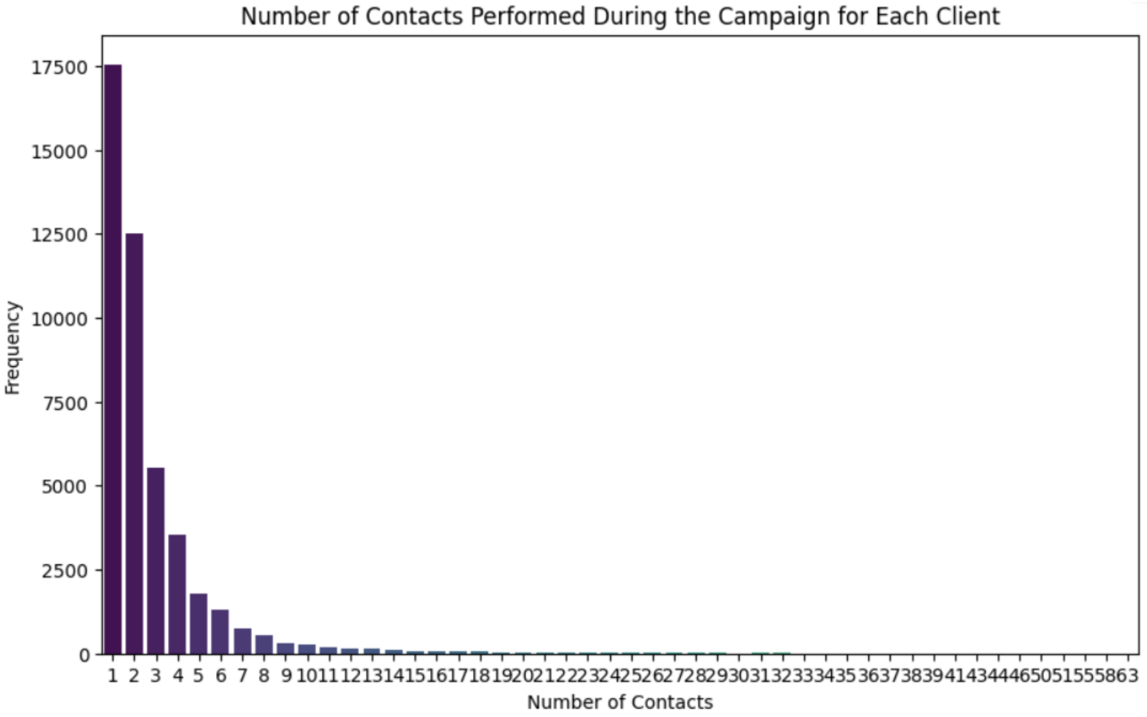
```
# What is the distribution of the duration of the last contact?
# Plotting the duration of the last contact distribution
plt.figure(figsize=(10, 6))
sns.histplot(data['duration'], bins=30, kde=True)
plt.title('Distribution of Duration of Last Contact')
plt.xlabel('Duration (seconds)')
plt.ylabel('Frequency')
plt.show()
```



Q.13 How many contacts were performed during the campaign for each client?

```
# How many contacts were performed during the campaign for each client?
# Plotting the number of contacts performed during the campaign
plt.figure(figsize=(10, 6))
sns.countplot(x=data['campaign'], palette='viridis')
plt.title('Number of Contacts Performed During the Campaign for Each Client')
plt.xlabel('Number of Contacts')
plt.ylabel('Frequency')
plt.show()

# Summary statistics of campaign contacts
campaign_contacts_summary = data['campaign'].describe()
print(campaign_contacts_summary)
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.


  sns.countplot(x=data['campaign'], palette='viridis')
```

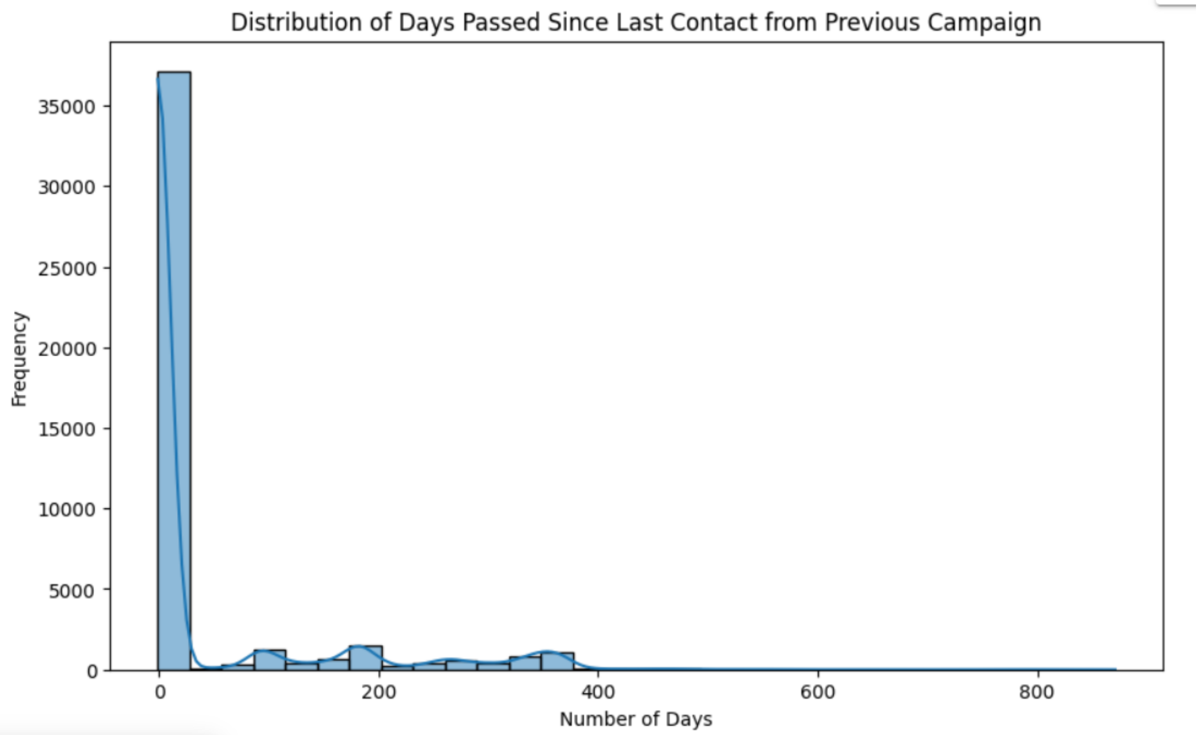


```
count    45216.000000
mean         2.763668
std          3.097896
min          1.000000
25%          1.000000
50%          2.000000
75%          3.000000
max         63.000000
Name: campaign, dtype: float64
```

Q.14 What is the distribution of the number of days passed since the client was last contacted from a previous campaign?

```
# What is the distribution of the number of days passed since the client was last
contacted from a previous campaign?
```

```
# Plotting the distribution of the number of days passed since the client was last contacted
plt.figure(figsize=(10, 6))
sns.histplot(data['pdays'], bins=30, kde=True)
plt.title('Distribution of Days Passed Since Last Contact from Previous Campaign')
plt.xlabel('Number of Days')
plt.ylabel('Frequency')
plt.show()

# Summary statistics of days passed since last contact
pdays_summary = data['pdays'].describe()
print(pdays_summary)
```



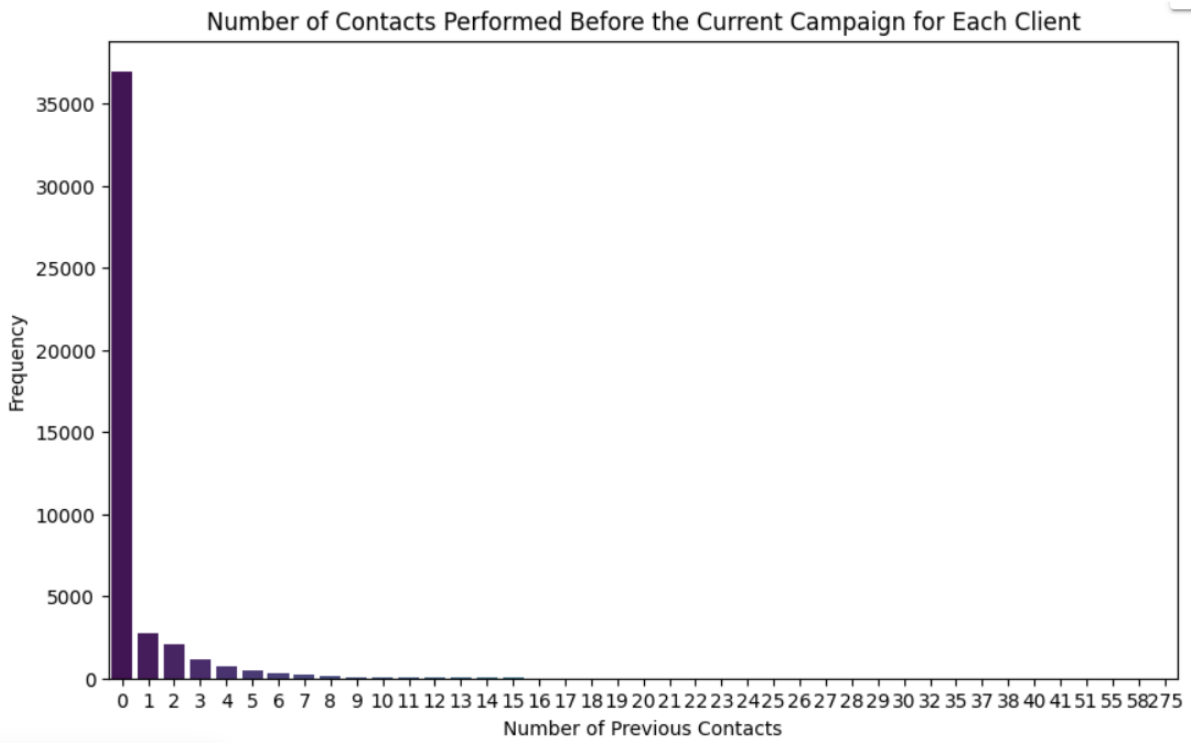
```
count    45216.000000
mean        40.202428
std        100.128248
min         -1.000000
25%         -1.000000
50%         -1.000000
```

```
75%        -1.000000
max       871.000000
Name: pdays, dtype: float64
```

Q.15 How many contacts were performed before the current campaign for each client?

```
# How many contacts were performed before the current campaign for each client?
# Plotting the number of contacts performed before the current campaign
plt.figure(figsize=(10, 6))
sns.countplot(x=data['previous'], palette='viridis')
plt.title('Number of Contacts Performed Before the Current Campaign for Each Client')
plt.xlabel('Number of Previous Contacts')
plt.ylabel('Frequency')
plt.show()

# Summary statistics of previous contacts
previous_contacts_summary = data['previous'].describe()
print(previous_contacts_summary)
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=data['previous'], palette='viridis')
```



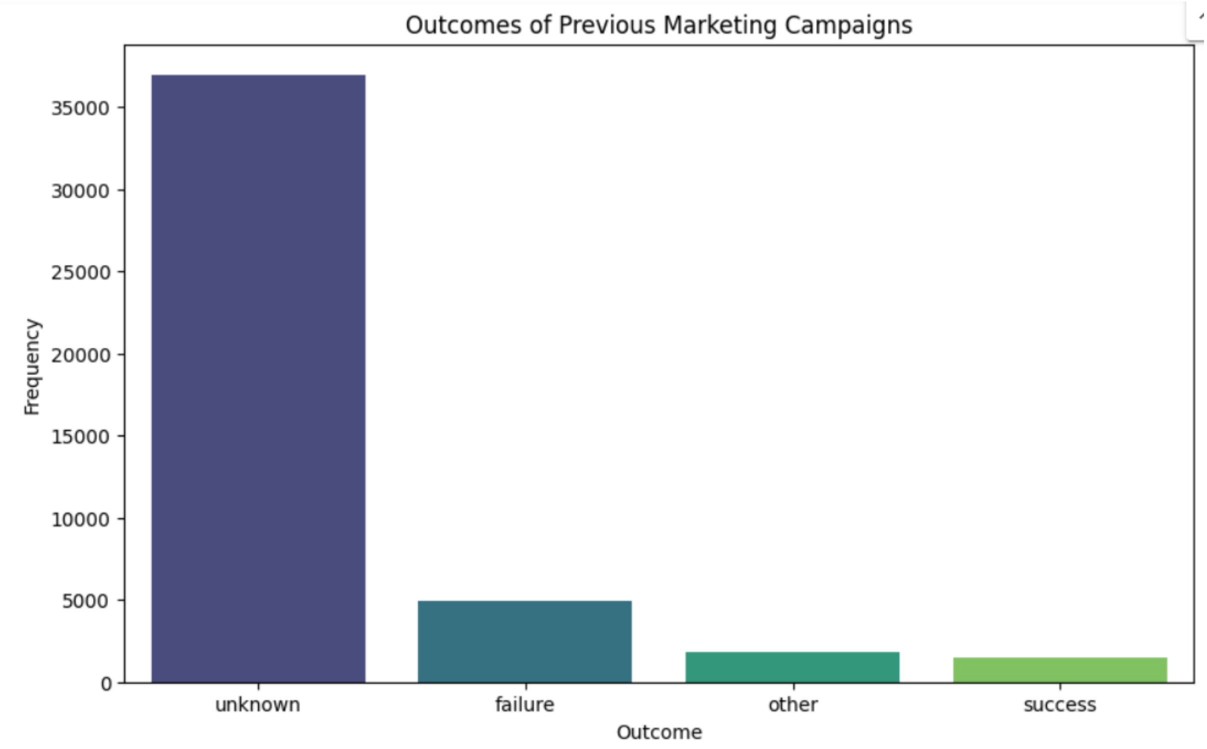
```
count    45216.000000
mean         0.580657
std          2.303778
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max        275.000000
Name: previous, dtype: float64
```

Q.16 What were the outcomes of the previous marketing campaigns?

```
# What were the outcomes of the previous marketing campaigns?
# Unique outcomes
previous_outcomes = data['poutcome'].unique()
print(previous_outcomes)

# Count of each outcome
```

```
previous_outcome_counts = data['poutcome'].value_counts()
print(previous_outcome_counts)

# Plotting the outcomes of previous marketing campaigns
plt.figure(figsize=(10, 6))
sns.countplot(x=data['poutcome'], palette='viridis')
plt.title('Outcomes of Previous Marketing Campaigns')
plt.xlabel('Outcome')
plt.ylabel('Frequency')
plt.show()
```

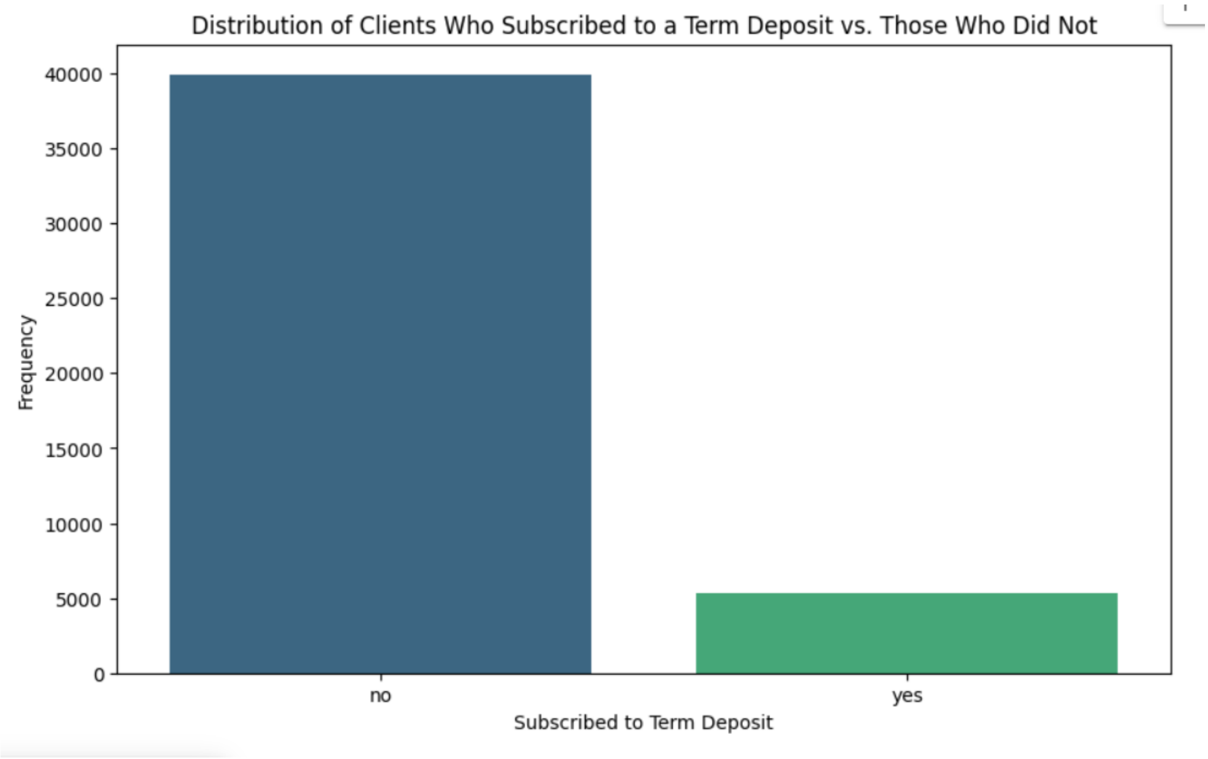
```
['unknown' 'failure' 'other' 'success']
poutcome
unknown    36961
failure     4902
other       1840
success     1513
Name: count, dtype: int64
<ipython-input-12-d9cd5046e36b>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=data['poutcome'], palette='viridis')
```



Q.17 What is the distribution of clients who subscribed to a term deposit vs. those who did not?

```
# What is the distribution of clients who subscribed to a term deposit vs. those who did not?

# Count of clients who subscribed vs. those who did not

subscription_counts = data['y'].value_counts()

print(subscription_counts)


# Plotting the distribution of clients who subscribed vs. those who did not

plt.figure(figsize=(10, 6))

sns.countplot(x=data['y'], palette='viridis')

plt.title('Distribution of Clients Who Subscribed to a Term Deposit vs. Those Who Did Not')

plt.xlabel('Subscribed to Term Deposit')

plt.ylabel('Frequency')

plt.show()
```

```
y
no     39922
yes     5294
Name: count, dtype: int64
<ipython-input-13-23ca6e5014e3>:8: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.


  sns.countplot(x=data['y'], palette='viridis')
```

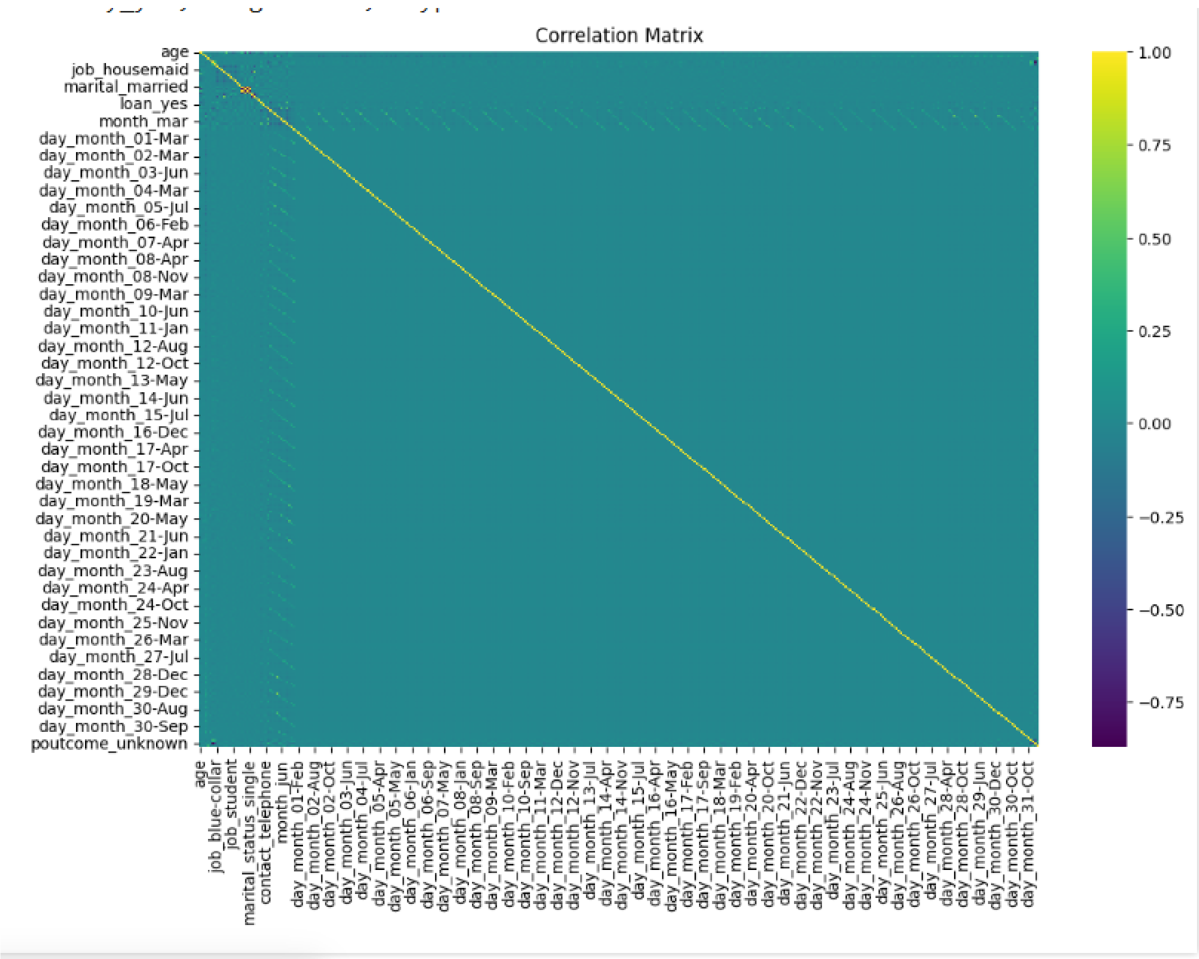


Q.18 Are there any correlations between different attributes and the likelihood of subscribing to a term deposit?

```
# Are there any correlations between different attributes and the likelihood of subscribing to a term deposit?

# Encoding categorical variables

data_encoded = pd.get_dummies(data, drop_first=True)


# Calculating the correlation matrix
```

```
correlation_matrix = data_encoded.corr()

# Extracting the correlation with the target variable 'y'
target_correlation = correlation_matrix['y_yes'].sort_values(ascending=False)
print(target_correlation)

# Visualizing the correlation with a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=False, cmap='viridis')
plt.title('Correlation Matrix')
plt.show()
```

```
y_yes               1.000000
duration            0.394387
poutcome_success    0.307083
month_mar           0.129371
month_oct           0.128439
                      ...
campaign           -0.073294
month_may          -0.102656
housing_yes        -0.139445
contact_unknown    -0.151062
poutcome_unknown   -0.167284
Name: y_yes, Length: 362, dtype: float64
```


Correlation Matrix
age
job_housemaid
marital_married
loan_yes
month_mar
day_month_01-Mar
day_month_02-Mar
day_month_03-Jun
day_month_04-Mar
day_month_05-Jul
day_month_06-Feb
day_month_07-Apr
day_month_08-Apr
day_month_08-Nov
day_month_09-Mar
day_month_10-Jun
day_month_11-Jan
day_month_12-Aug
day_month_12-Oct
day_month_13-May
day_month_14-Jun
day_month_15-Jul
day_month_16-Dec
day_month_17-Apr
day_month_17-Oct
day_month_18-May
day_month_19-Mar
day_month_20-May
day_month_21-Jun
day_month_22-Jan
day_month_23-Aug
day_month_24-Apr
day_month_24-Oct
day_month_25-Nov
day_month_26-Mar
day_month_27-Jul
day_month_28-Dec
day_month_29-Dec
day_month_30-Aug
day_month_30-Sep
poutcome_unknown
age
job_blue-collar
job_student
marital_status_single
contact_telephone
month_jun
day_month_01-Feb
day_month_02-Aug
day_month_02-Oct
day_month_03-Jun
day_month_04-Jul
day_month_05-Apr
day_month_05-May
day_month_06-Jan
day_month_06-Sep
day_month_07-May
day_month_08-Jan
day_month_08-Sep
day_month_09-Mar
day_month_10-Feb
day_month_10-Sep
day_month_11-Mar
day_month_12-Dec
day_month_12-Nov
day_month_13-Jul
day_month_14-Apr
day_month_14-Nov
day_month_15-Jul
day_month_16-Apr
day_month_16-May
day_month_17-Feb
day_month_17-Sep
day_month_18-Mar
day_month_19-Feb
day_month_20-Apr
day_month_20-Oct
day_month_21-Jun
day_month_22-Dec
day_month_22-Nov
day_month_23-Jul
day_month_24-Aug
day_month_24-Nov
day_month_25-Jun
day_month_26-Aug
day_month_26-Oct
day_month_27-Jul
day_month_28-Apr
day_month_28-Oct
day_month_29-Jun
day_month_30-Dec
day_month_31-Oct
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75