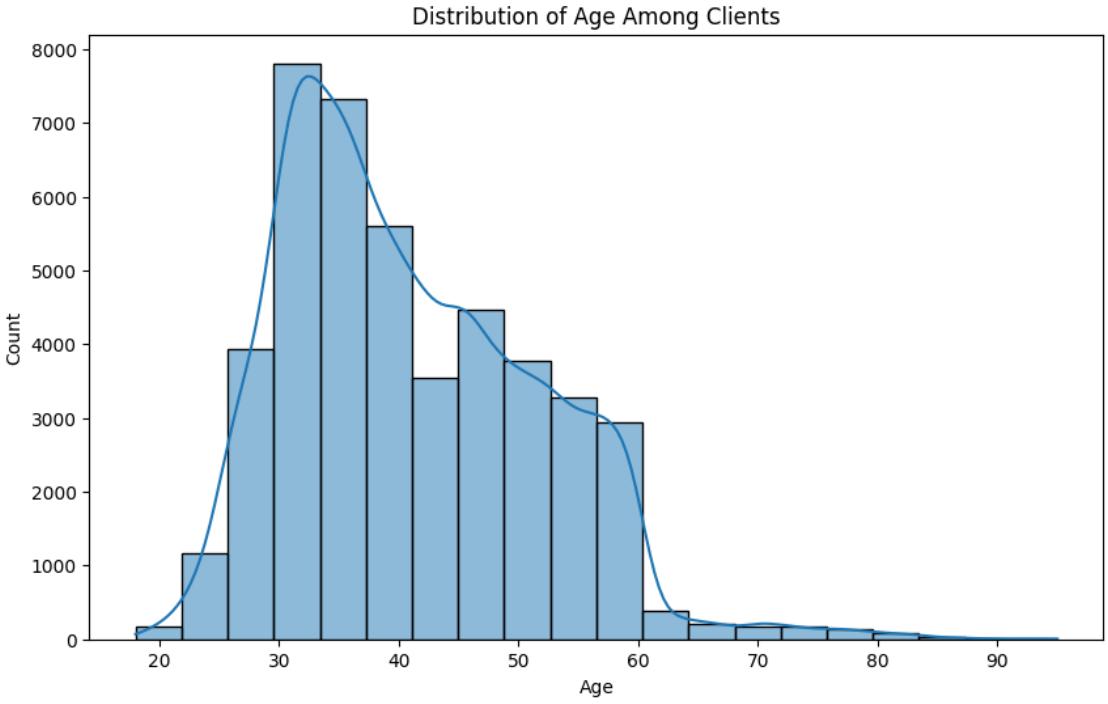
devanshu-finlatics-project

April 18, 2024

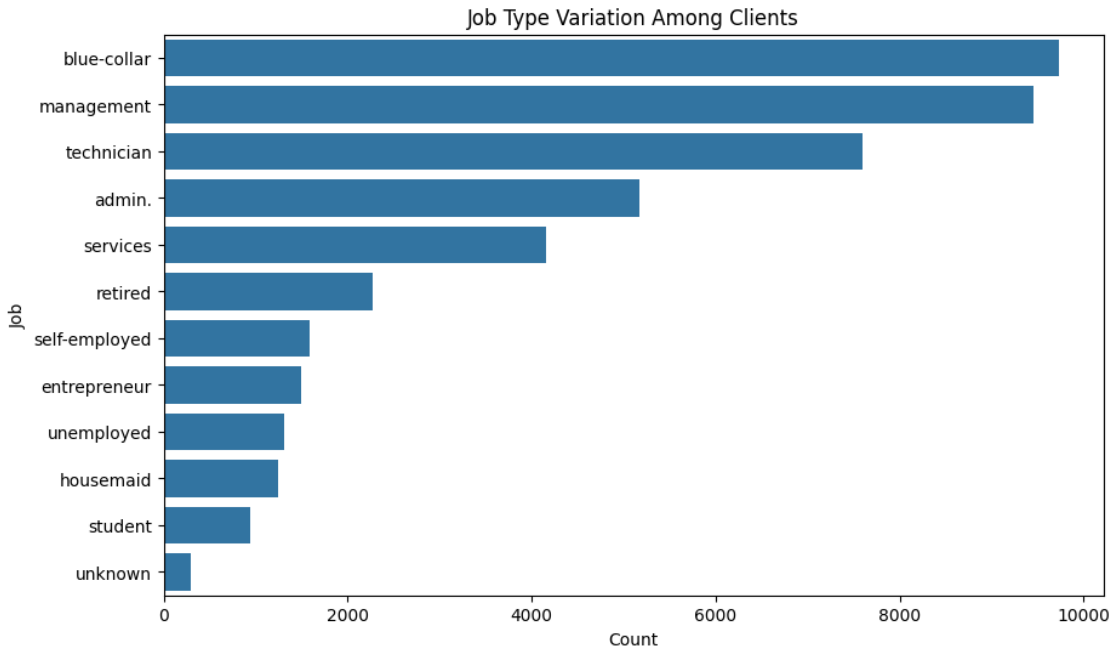
[26]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv("banking_data.csv")
```

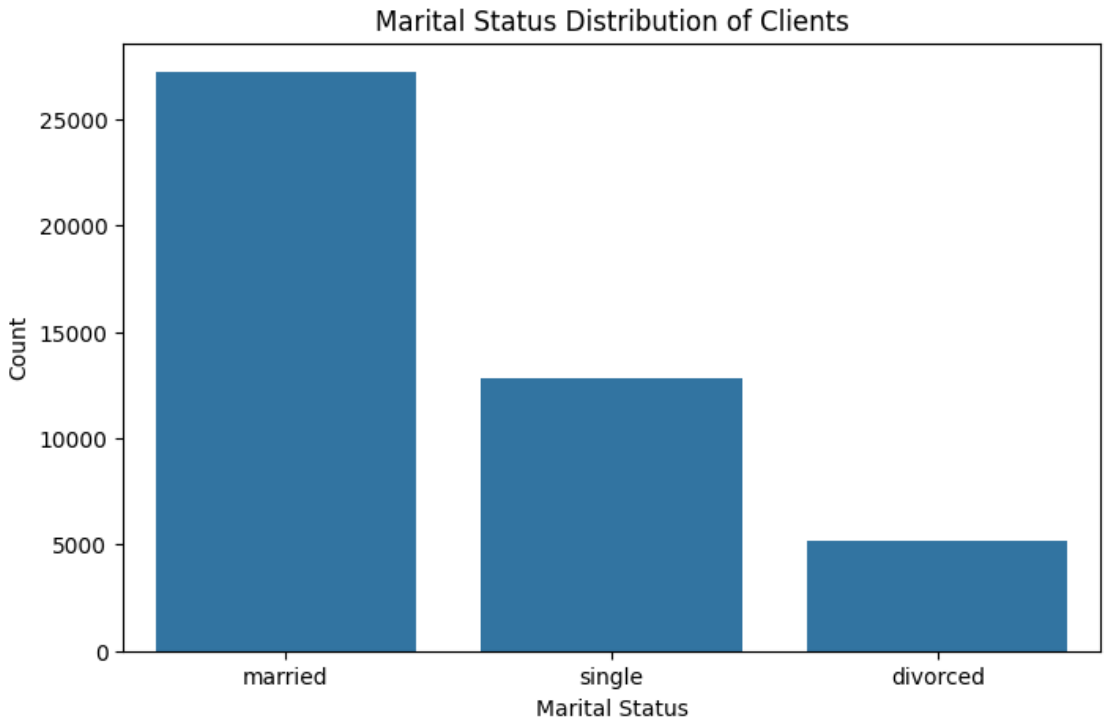
[27]:
```
#1
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='age', bins=20, kde=True)
plt.title('Distribution of Age Among Clients')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



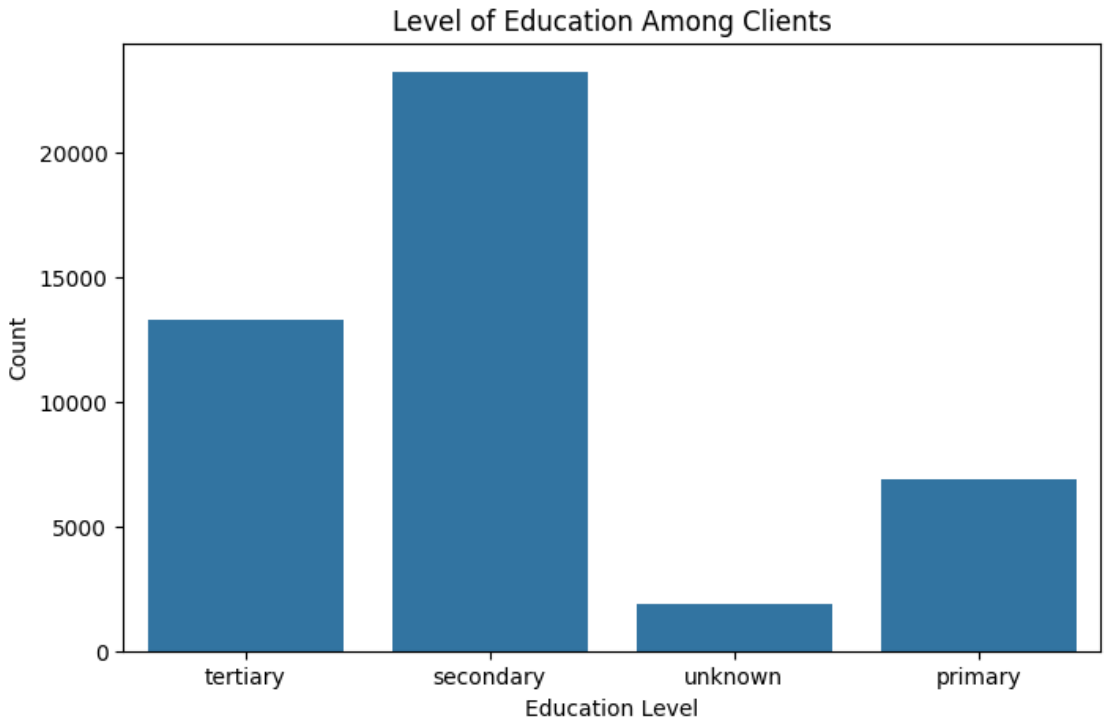
[28]:
```
#2
plt.figure(figsize=(10, 6))
sns.countplot(data=data, y='job', order=data['job'].value_counts().index)
plt.title('Job Type Variation Among Clients')
plt.xlabel('Count')
plt.ylabel('Job')
plt.show()
```



[29]:
```
#3
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='marital')
plt.title('Marital Status Distribution of Clients')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()
```



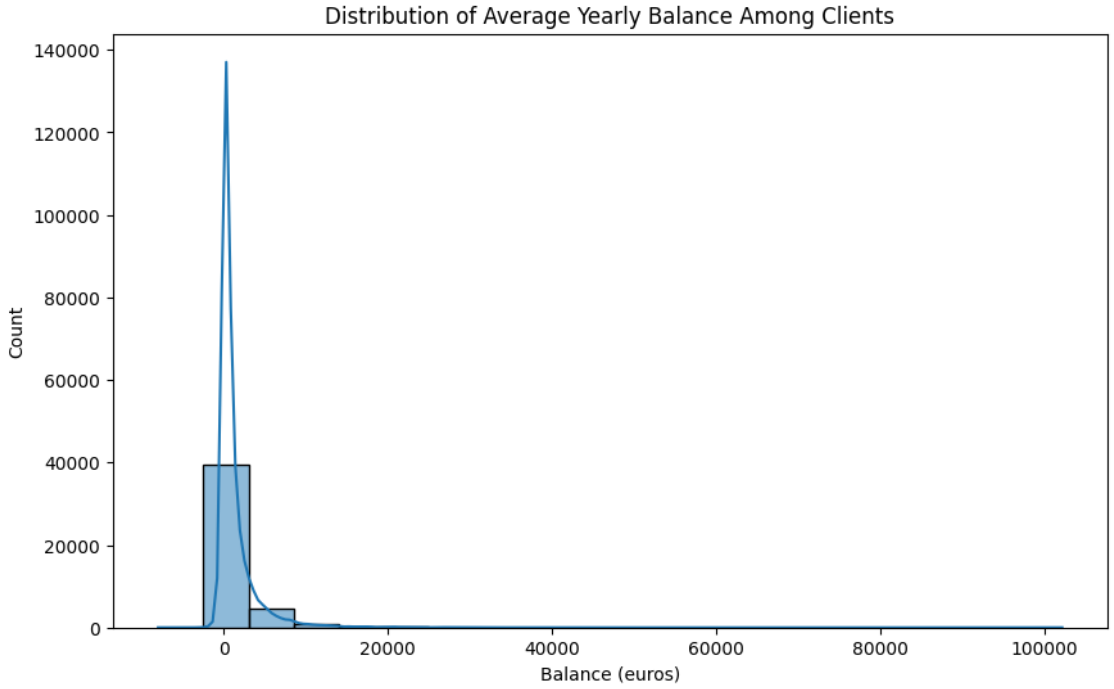
[30]:
```
#4
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='education')
plt.title('Level of Education Among Clients')
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.show()
```



[31]:
```
#5
default_counts = data['default'].value_counts(normalize=True) * 100
print("Proportion of clients with credit in default:")
print(default_counts)
```

```
Proportion of clients with credit in default:
default
no     98.197541
yes     1.802459
Name: proportion, dtype: float64
```

[32]:
```
#6
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='balance', bins=20, kde=True)
plt.title('Distribution of Average Yearly Balance Among Clients')
plt.xlabel('Balance (euros)')
plt.ylabel('Count')
plt.show()
```



[33]:
```
#7
housing_counts = data['housing'].value_counts()
print("Number of clients with housing loans:")
print(housing_counts)
```

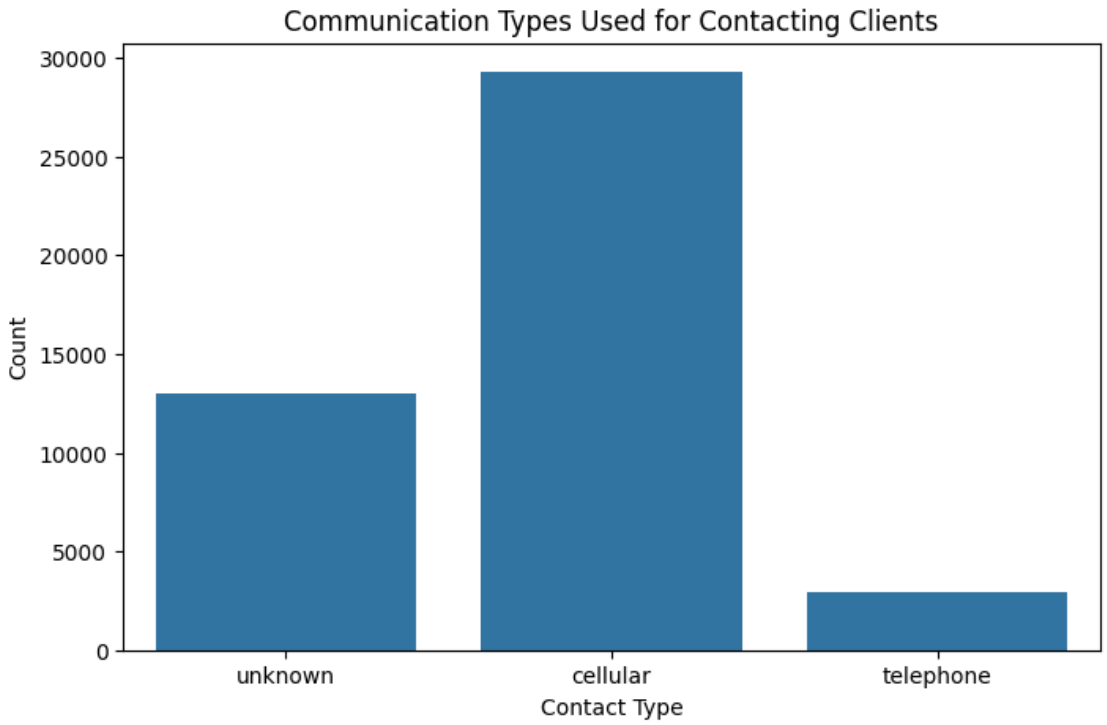
```
Number of clients with housing loans:
housing
yes    25130
no     20086
Name: count, dtype: int64
```

[34]:
```
#8
loan_counts = data['loan'].value_counts()
print("Number of clients with personal loans:")
print(loan_counts)
```

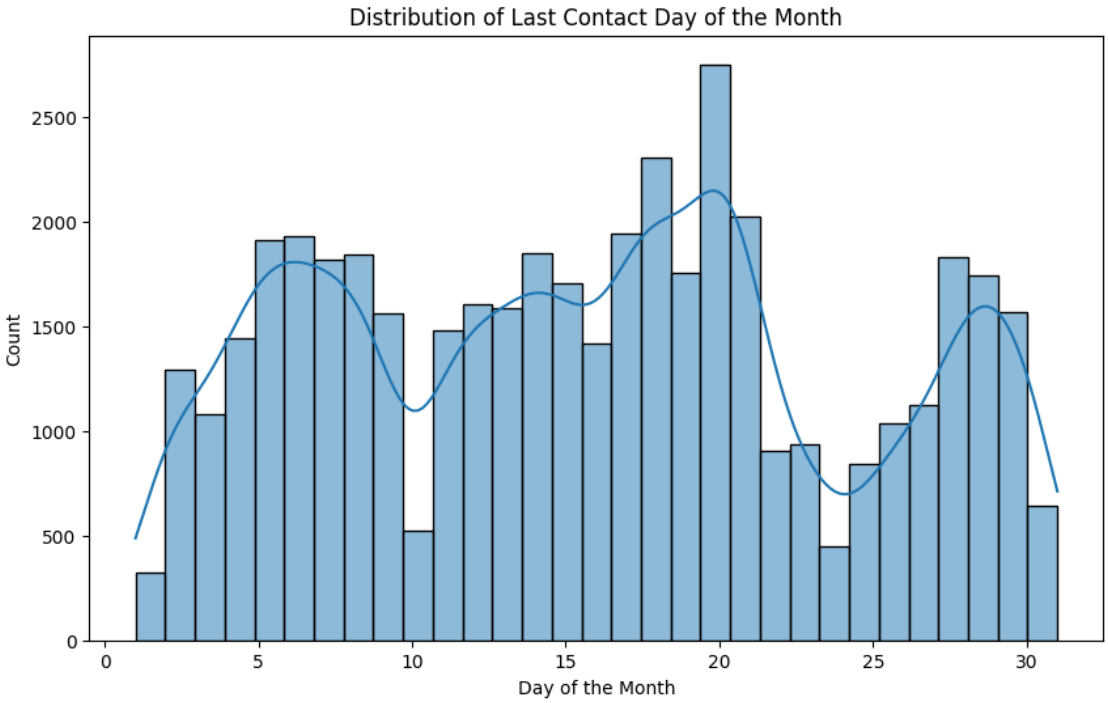
```
Number of clients with personal loans:
loan
no     37972
yes     7244
Name: count, dtype: int64
```

[35]:
```
#9
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='contact')
```

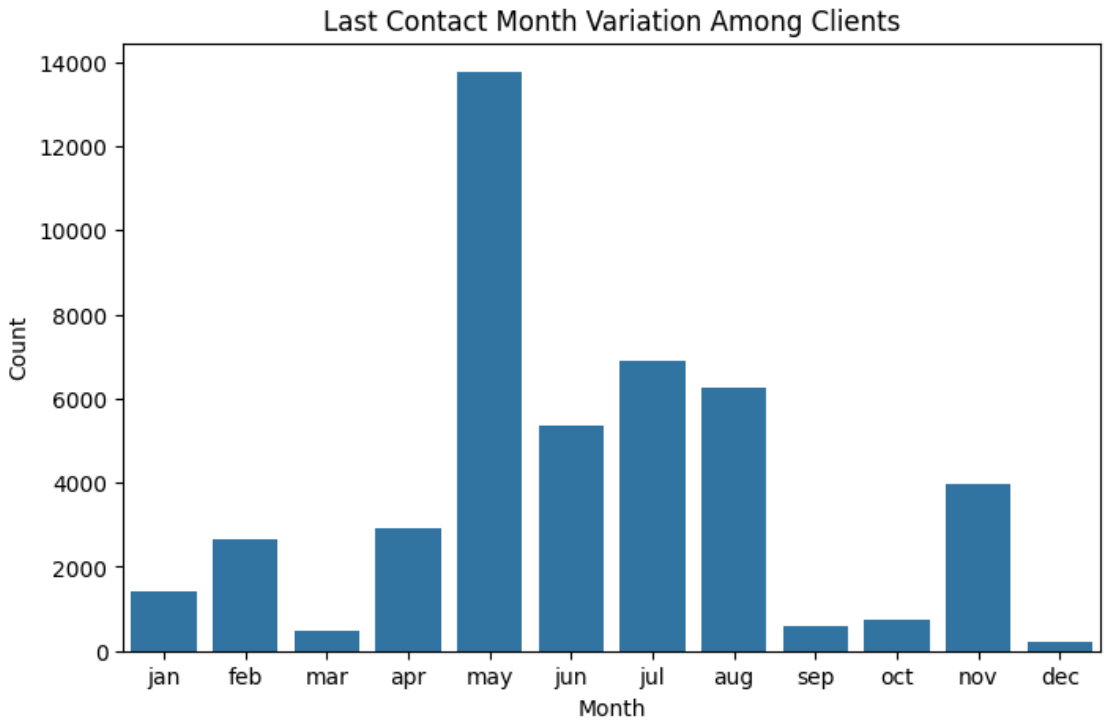
```
plt.title('Communication Types Used for Contacting Clients')
plt.xlabel('Contact Type')
plt.ylabel('Count')
plt.show()
```



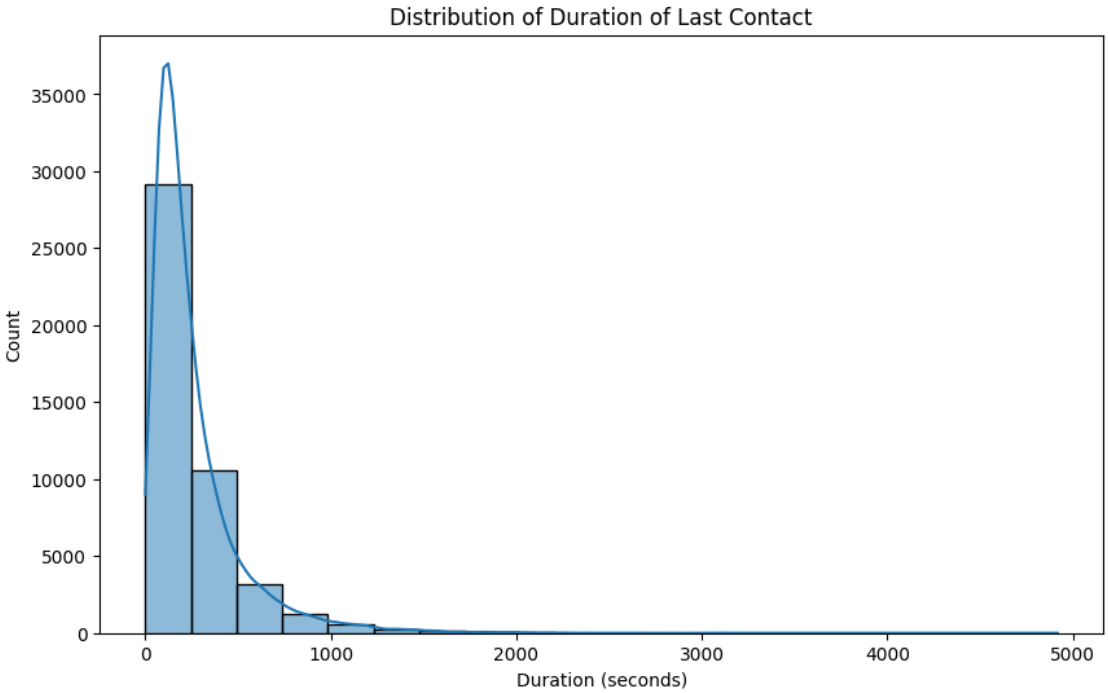
[36]:
```
#10
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='day', bins=31, kde=True)
plt.title('Distribution of Last Contact Day of the Month')
plt.xlabel('Day of the Month')
plt.ylabel('Count')
plt.show()
```



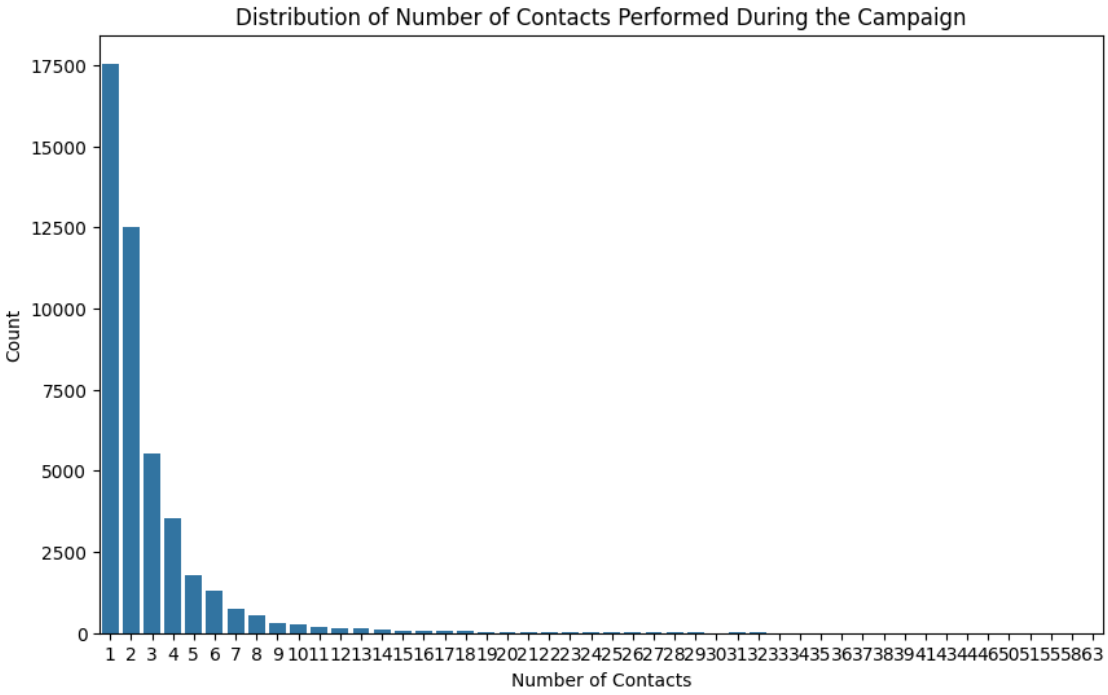
[37]:
```
#11
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='month', order=['jan', 'feb', 'mar', 'apr', 'may',␣
 ↪'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec'])
plt.title('Last Contact Month Variation Among Clients')
plt.xlabel('Month')
plt.ylabel('Count')
plt.show()
```



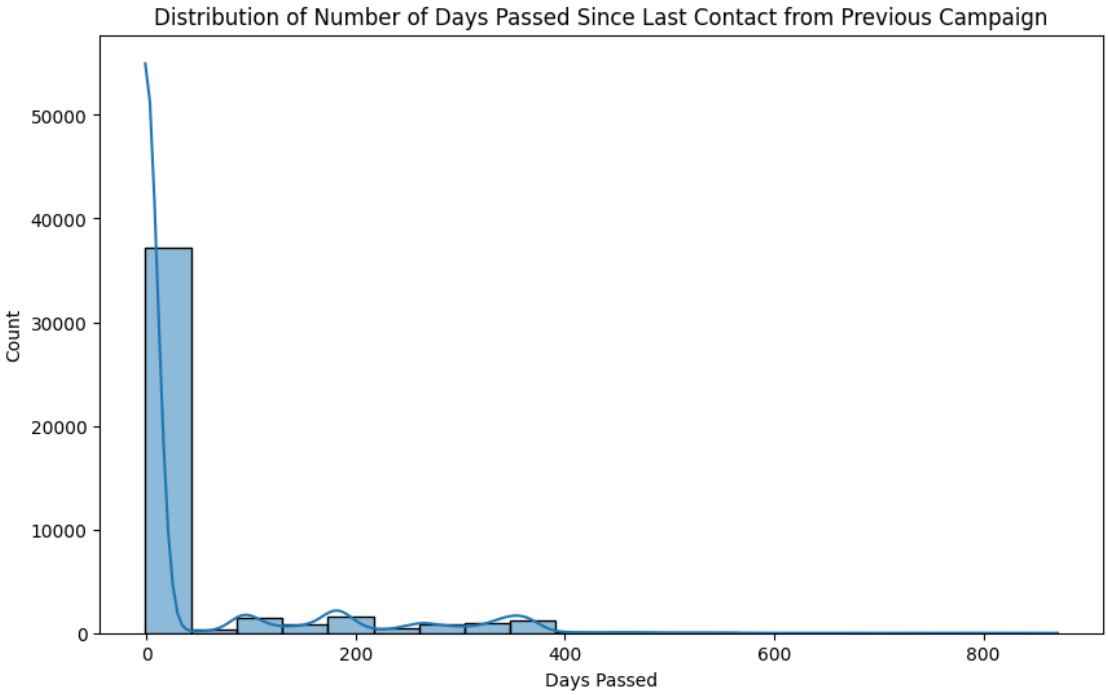
[38]:
```
#12
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='duration', bins=20, kde=True)
plt.title('Distribution of Duration of Last Contact')
plt.xlabel('Duration (seconds)')
plt.ylabel('Count')
plt.show()
```



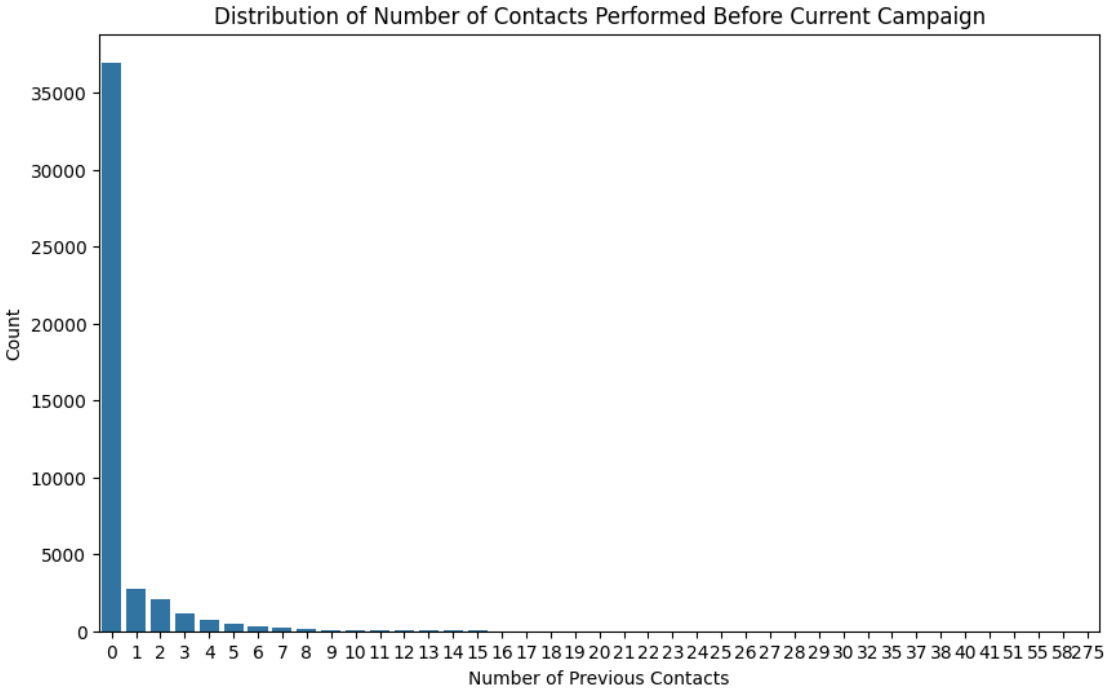
[39]:
```
#13
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='campaign')
plt.title('Distribution of Number of Contacts Performed During the Campaign')
plt.xlabel('Number of Contacts')
plt.ylabel('Count')
plt.show()
```



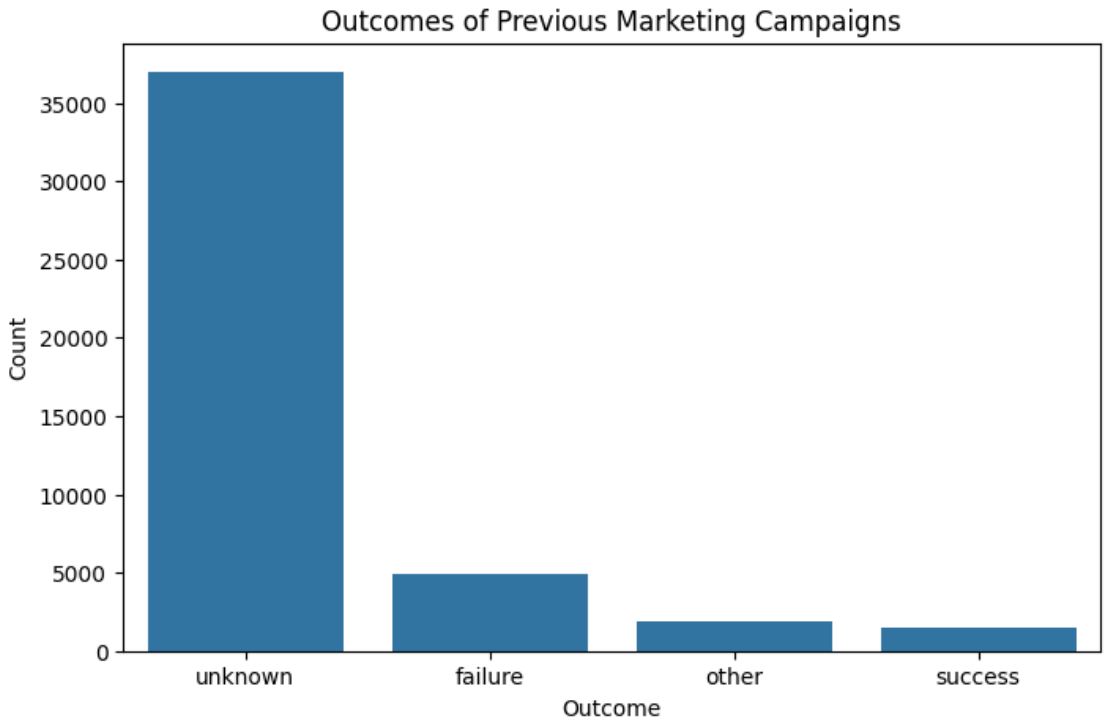
[40]:
```
#14
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='pdays', bins=20, kde=True)
plt.title('Distribution of Number of Days Passed Since Last Contact from␣
 ↪Previous Campaign')
plt.xlabel('Days Passed')
plt.ylabel('Count')
plt.show()
```



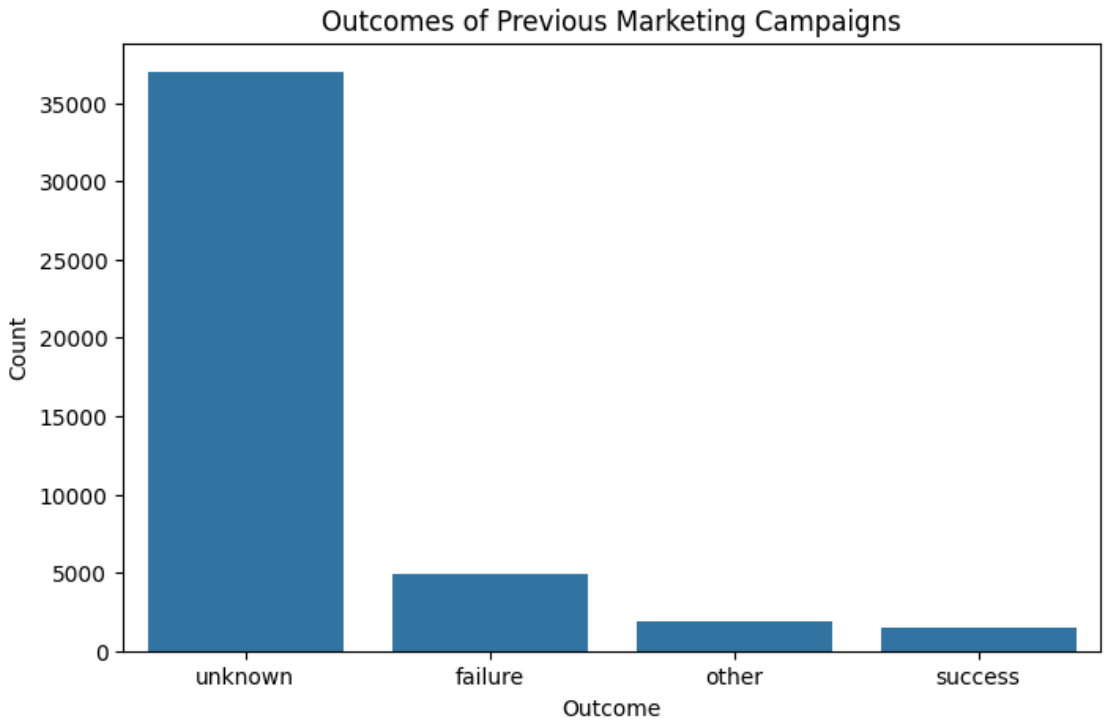
[41]:
```
#15
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='previous')
plt.title('Distribution of Number of Contacts Performed Before Current␣
 ↪Campaign')
plt.xlabel('Number of Previous Contacts')
plt.ylabel('Count')
plt.show()
```



[42]:
```
#16
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='poutcome')
plt.title('Outcomes of Previous Marketing Campaigns')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.show()
```



[43]:
```
#17
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='poutcome')
plt.title('Outcomes of Previous Marketing Campaigns')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.show()
```



[46]:
```
#18
# Correlation between attributes and likelihood of subscribing to a term deposit
plt.figure(figsize=(12, 8))
correlation_with_y = data_encoded.corr()['y'].sort_values(ascending=False)
sns.heatmap(data_encoded[correlation_with_y.index].corr(), annot=True,␣
 ↪cmap='coolwarm', fmt=".2f")
plt.title('Correlation between Attributes and Likelihood of Subscribing to a␣
 ↪Term Deposit')
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-fe49d5cf45bf> in <cell line: 4>()
      2 # Correlation between attributes and likelihood of subscribing to a ter␣
 ↪deposit
      3 plt.figure(figsize=(12, 8))
----> 4 correlation_with_y = data_encoded.corr()['y'].
 ↪sort_values(ascending=False)
      5 sns.heatmap(data_encoded[correlation_with_y.index].corr(), annot=True,␣
 ↪cmap='coolwarm', fmt=".2f")
      6 plt.title('Correlation between Attributes and Likelihood of Subscribing␣
 ↪to a Term Deposit')
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in corr(self,␣
 ↪method, min_periods, numeric_only)
  10052         cols = data.columns
  10053         idx = cols.copy()
> 10054         mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10055
  10056         if method == "pearson":

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in to_numpy(self,␣
 ↪dtype, copy, na_value)
   1836         if dtype is not None:
   1837             dtype = np.dtype(dtype)
-> 1838         result = self._mgr.as_array(dtype=dtype, copy=copy,␣
 ↪na_value=na_value)
   1839         if result.dtype is not dtype:
   1840             result = np.array(result, dtype=dtype, copy=False)

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in␣
 ↪as_array(self, dtype, copy, na_value)
   1730                 arr.flags.writeable = False
   1731         else:
-> 1732             arr = self._interleave(dtype=dtype, na_value=na_value)
   1733             # The underlying data was copied within _interleave, so no␣
 ↪need
   1734             # to further copy if copy=True or setting na_value

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in␣
 ↪_interleave(self, dtype, na_value)
   1792             else:
   1793                 arr = blk.get_values(dtype)
-> 1794             result[rl.indexer] = arr
   1795             itemmask[rl.indexer] = 1
   1796

ValueError: could not convert string to float: 'married'
```

<Figure size 1200x800 with 0 Axes>