Marketing Campaign Analysis

Load the Data set

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

df = pd.read_excel('/content/marketing_campaign.xlsx')

df.head(5)
```

→		ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custome
	0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-0
	1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-0
	2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-2
	3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-1
	4	5324	1981	PhD	Married	58293.0	1	0	2014-01-1

5 rows × 29 columns

Data Cleaning

Check for missing values

missing_values = df.isnull().sum()

Fill missing values in 'Income' with the median value

```
df['Income'].fillna(df['Income'].median(), inplace=True)
```

<ipython-input-12-b0f8a841459e>:1: FutureWarning: A value is trying to be set on a cc The behavior will change in pandas 3.0. This inplace method will never work because t

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({

df.info()

```
df['Income'].fillna(df['Income'].median(), inplace=True)
```

Convert 'Dt Customer' to datetime format

```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%Y-%m-%d')
```

Drop any rows with missing values in critical columns

```
df.dropna(subset=['Income', 'Education', 'Marital_Status'], inplace=True)
```

Verify the data after cleaning

14 MntGoldProds

18 NumStorePurchases

19 NumWebVisitsMonth

20 AcceptedCmp3

21 AcceptedCmp4

23 AcceptedCmp1

24 AcceptedCmp2

Z CostContact

25 Complain

22 AcceptedCmp5

17

NumDealsPurchasesNumWebPurchases2240 non-null2240 non-null

NumCatalogPurchases 2240 non-null

```
<<class 'pandas.core.frame.DataFrame'>
     RangeIndex: 2240 entries, 0 to 2239
     Data columns (total 29 columns):
           Column
                                      Non-Null Count Dtype
           ----
                                      -----
      0
           ID
                                     2240 non-null
                                                          int64
          Year_Birth
                                     2240 non-null int64
      1
           Education
                                     2240 non-null object
         Marital_Status
      3
                                   2240 non-null object
                                     2240 non-null
          Income
                                                          float64
                                    2240 non-null
           Kidhome
                                                           int64
      6 Teenhome 2240 non-null
7 Dt_Customer 2240 non-null
8 Recency 2240 non-null
9 MntWines 2240 non-null
10 MntFruits 2240 non-null
11 MntMeatProducts 2240 non-null
12 MntFishProducts 2240 non-null
13 MntSweetProducts 2240 non-null
                                                           int64
                                                           datetime64[ns]
                                                           int64
                                                           int64
                                                           int64
                                                           int64
                                                           int64
      13 MntSweetProducts
                                    2240 non-null
                                                           int64
                                   2240 non-null
```

2240 non-null

int64

int64 int64

int64

int64

int64

int64

int64

int64

int64

int64

int64

int64

```
27 Z_Revenue 2240 non-null int64
28 Response 2240 non-null int64
dtypes: datetime64[ns](1), float64(1), int64(25), object(2)
memory usage: 507.6+ KB
```

Ensuring Data Consistency

Check unique values in 'Marital_Status' to spot inconsistencies

```
print(df['Marital_Status'].unique())
df['Marital_Status'] = df['Marital_Status'].replace(['Alone'], 'Single')
df['Marital_Status'] = df['Marital_Status'].str.lower()

Type ['Single' 'Together' 'Married' 'Divorced' 'Widow' 'Alone' 'Absurd' 'YOLO']
```

Standardizing Data Formats

Convert 'Dt_Customer' to datetime format

```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%Y-%m-%d')
```

Standardize numerical columns

```
df['Income'] = df['Income'].astype(float)
```

Convert categorical columns to lowercase and ensure consistency

```
df['Education'] = df['Education'].str.lower()
df['Marital_Status'] = df['Marital_Status'].str.lower()
```

Verify the standardized formats

print(df.dtypes)

→	ID	int64
	Year_Birth	int64
	Education	object
	Marital_Status	object
	Income	float64
	Kidhome	int64
	Teenhome	int64
	Dt_Customer	datetime64[ns]
	Recency	int64
	MntWines	int64
	MntFruits	int64
	MntMeatProducts	int64
	MntFishProducts	int64
	MntSweetProducts	int64
	MntGoldProds	int64
	NumDealsPurchases	int64
	NumWebPurchases	int64
	NumCatalogPurchases	int64
	NumStorePurchases	int64
	NumWebVisitsMonth	int64
	AcceptedCmp3	int64
	AcceptedCmp4	int64
	AcceptedCmp5	int64
	AcceptedCmp1	int64
	AcceptedCmp2	int64
	Complain	int64
	<pre>Z_CostContact</pre>	int64
	Z_Revenue	int64
	_ Response	int64
	dtype: object	

Data Analysis

Calculate Total Revenue from marketing campaigns

```
df['Total_Revenue'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntF
```

Calculate ROI for campaigns

```
df['ROI'] = (df['Z_Revenue'] - df['Z_CostContact']) / df['Z_CostContact']
```

Calculate Cost Per Acquisition (CPA)

df['CPA'] = df['Z_CostContact'] / (df['AcceptedCmp1'] + df['AcceptedCmp2'] + df['Accepted

Analyze key metrics

campaign_performance = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4'

/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:4655: RuntimeWarni diff_b_a = subtract(b, a)

Display the performance

campaign_performance

$\overline{}$							
→		AcceptedCmp1	AcceptedCmp2	AcceptedCmp3	AcceptedCmp4	AcceptedCmp5	
	count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2.240000
	mean	0.064286	0.013393	0.072768	0.074554	0.072768	2.6666670
	std	0.245316	0.114976	0.259813	0.262728	0.259813	5.374679
	min	0.000000	0.000000	0.000000	0.000000	0.000000	2.6666670
	25%	0.000000	0.000000	0.000000	0.000000	0.000000	2.6666670
	50%	0.000000	0.000000	0.000000	0.000000	0.000000	2.6666670
	75%	0.000000	0.000000	0.000000	0.000000	0.000000	2.6666670
	max	1.000000	1.000000	1.000000	1.000000	1.000000	2.666667
	4						>

Analyzing Campaign Performance Based on Key Metrics

Analyze the number of accepted campaigns

campaign_acceptances = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4'
print("Number of customers who accepted each campaign:\n", campaign_acceptances)

Number of customers who accepted each campaign:
AcceptedCmp1 144

```
AcceptedCmp2 30
AcceptedCmp3 163
AcceptedCmp4 167
AcceptedCmp5 163
dtype: int64
```

Analyze the overall response rate

```
response_rate = df['Response'].value_counts(normalize=True) * 100
print("Response rate (%):\n", response_rate)

Response rate (%):
    Response
    0   85.089286
    1   14.910714
    Name: proportion, dtype: float64
```

Check the recency of campaigns

```
print("Recency stats (days since last interaction):")
print(df['Recency'].describe())
    Recency stats (days since last interaction):
    count 2240.000000
    mean
              49.109375
              28.962453
    std
    min
               0.000000
    25%
              24.000000
    50%
              49.000000
    75%
              74.000000
    max
               99.000000
    Name: Recency, dtype: float64
```

Calculating ROI and Cost Per Acquisition (CPA)

Calculate total revenue from campaigns

```
df['Total_Revenue'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntF
```

Calculate ROI for each campaign

```
df['ROI'] = (df['Total_Revenue'] - df['Z_CostContact']) / df['Z_CostContact']
```

Calculate Cost Per Acquisition (CPA)

```
df['Total_Accepted'] = df['AcceptedCmp1'] + df['AcceptedCmp2'] + df['AcceptedCmp3'] + df[
```

CPA = Total Cost / Number of Acceptances

```
df['CPA'] = df['Z_CostContact'] / df['Total_Accepted']
```

Display the average ROI and CPA

```
print("Average ROI: ", df['ROI'].mean())
print("Average CPA: ", df['CPA'].mean())

Average ROI: 200.93273809523808
Average CPA: inf
```

Identifying Factors Contributing to Successful Campaigns

Analyze correlations between key variables and the

'Response'

```
numerical_features = df.select_dtypes(include=np.number).columns
correlation_matrix = df[numerical_features].corr()
print("Correlation between different features:\n", correlation_matrix['Response'].sort_va
```

Correlation between different features:

Response	1.000000
Total_Accepted	0.426035
AcceptedCmp5	0.326634
AcceptedCmp1	0.293982
ROI	0.265298
Total_Revenue	0.265298
AcceptedCmp3	0.254258
MntWines	0.247254

```
MntMeatProducts
                       0.236335
NumCatalogPurchases
                       0.220810
AcceptedCmp4
                       0.177019
AcceptedCmp2
                       0.169293
NumWebPurchases
                       0.148730
MntGoldProds
                       0.139850
Income
                       0.132867
MntFruits
                       0.125289
MntSweetProducts
                       0.117372
MntFishProducts
                       0.111331
NumStorePurchases
                       0.039363
Year_Birth
                       0.021325
NumDealsPurchases
                       0.002238
Complain
                      -0.001707
NumWebVisitsMonth
                      -0.003987
ID
                      -0.021968
Kidhome
                      -0.080008
Teenhome
                      -0.154446
Recency
                      -0.198437
                      -0.324961
CPA
Z_CostContact
                            NaN
Z Revenue
Name: Response, dtype: float64
```

Group by marital status and calculate the average response rate

```
marital_response = df.groupby('Marital_Status')['Response'].mean()
print("Response rate by marital status:\n", marital_response)
    Response rate by marital status:
      Marital Status
     absurd
                0.500000
     divorced
                 0.206897
     married
                0.113426
                 0.221532
     single
     together
                 0.103448
     widow
                 0.246753
```

0.500000

Name: Response, dtype: float64

yolo

Group by education level and calculate the average response rate

```
education_response = df.groupby('Education')['Response'].mean()
print("Response rate by education level:\n", education_response)

Response rate by education level:
    Education
```

```
2n Cycle 0.108374
Basic 0.037037
Graduation 0.134871
Master 0.154054
PhD 0.207819
Name: Response, dtype: float64
```

Analyze impact of income on response rates

```
income_response = df.groupby(pd.cut(df['Income'], bins=5))['Response'].mean()
print("Response rate by income level:\n", income_response)
Response rate by income level:
      Income
     (1065.064, 134717.2]
                            0.150815
     (134717.2, 267704.4]
                            0.000000
     (267704.4, 400691.6]
                                 NaN
     (400691.6, 533678.8]
                                 NaN
     (533678.8, 666666.0] 0.000000
     Name: Response, dtype: float64
     <ipython-input-39-099a47d32894>:1: FutureWarning: The default of observed=False is de
       income_response = df.groupby(pd.cut(df['Income'], bins=5))['Response'].mean()
```

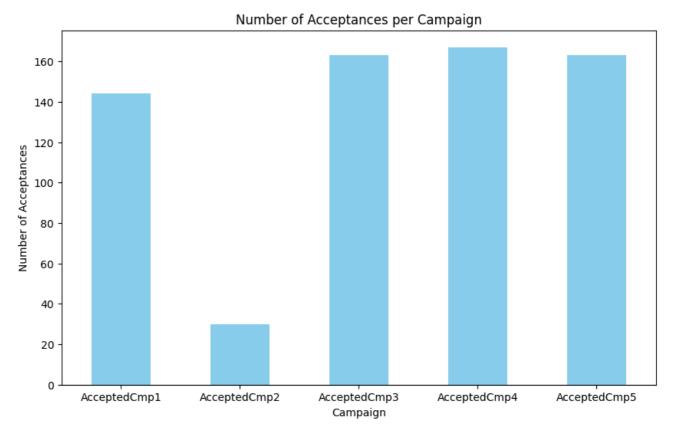
Visualization

Bar Charts for Campaign Acceptance

```
accepted_cmp = ['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'Accepted
campaign_acceptances = df[accepted_cmp].sum()

plt.figure(figsize=(10, 6))
campaign_acceptances.plot(kind='bar', color='skyblue')
plt.title('Number of Acceptances per Campaign')
plt.xlabel('Campaign')
plt.ylabel('Number of Acceptances')
plt.xticks(rotation=0)
plt.show()
```

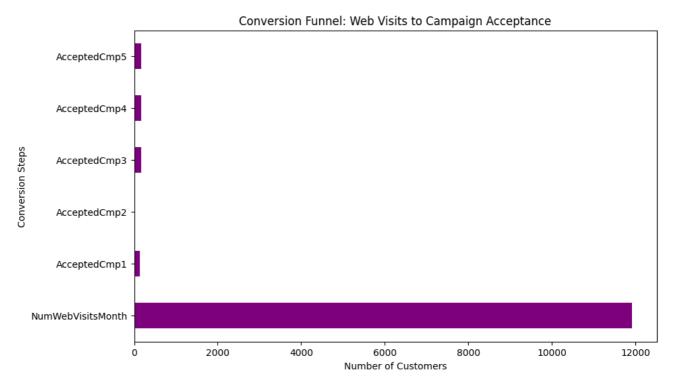
 $\overline{2}$



Conversion Funnel

```
conversion_steps = df[['NumWebVisitsMonth', 'AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3']
plt.figure(figsize=(10, 6))
conversion_steps.plot(kind='barh', color='purple')
plt.title('Conversion Funnel: Web Visits to Campaign Acceptance')
plt.xlabel('Number of Customers')
plt.ylabel('Conversion Steps')
plt.show()
```

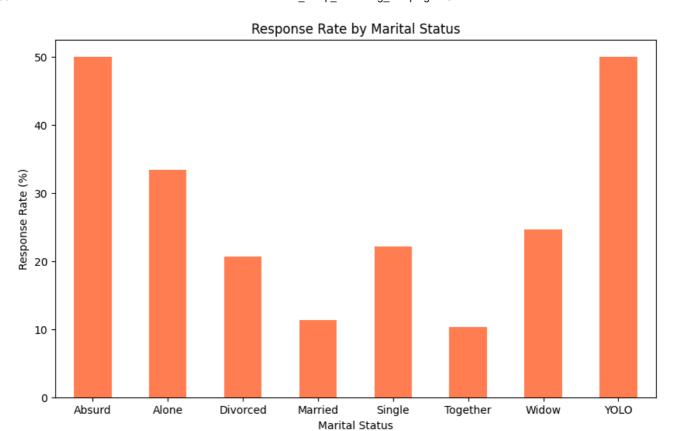
 $\overline{2}$



Bar Chart for Response Rate by Marital Status

```
marital_response = df.groupby('Marital_Status')['Response'].mean() * 100
plt.figure(figsize=(10, 6))
marital_response.plot(kind='bar', color='coral')
plt.title('Response Rate by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Response Rate (%)')
plt.xticks(rotation=0)
plt.show()
```

 $\overline{\mathbf{T}}$



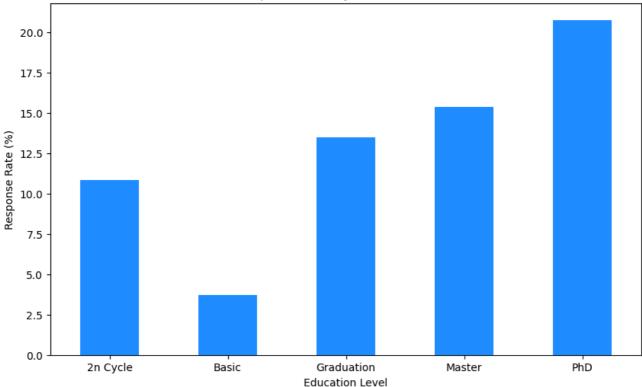
Bar Chart for Response Rate by Education Level

```
education_response = df.groupby('Education')['Response'].mean() * 100

plt.figure(figsize=(10, 6))
education_response.plot(kind='bar', color='dodgerblue')
plt.title('Response Rate by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Response Rate (%)')
plt.xticks(rotation=0)
plt.show()
```

 $\overline{\mathbf{T}}$



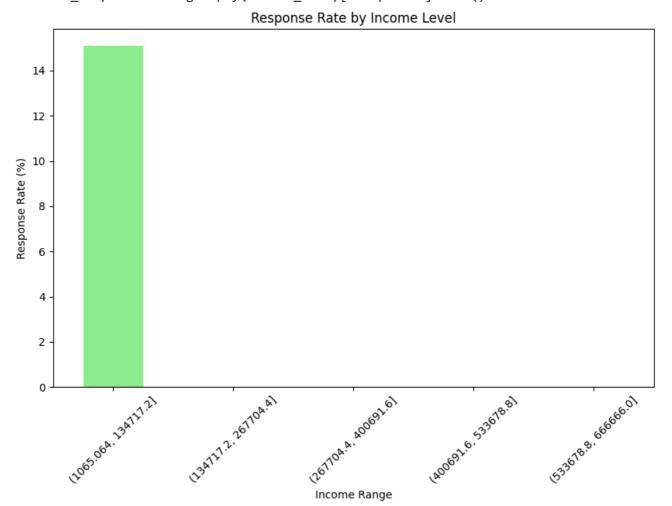


Bar Chart for Response Rate by Income Level

```
income_bins = pd.cut(df['Income'], bins=5)
income_response = df.groupby(income_bins)['Response'].mean() * 100

plt.figure(figsize=(10, 6))
income_response.plot(kind='bar', color='lightgreen')
plt.title('Response Rate by Income Level')
plt.xlabel('Income Range')
plt.ylabel('Response Rate (%)')
plt.xticks(rotation=45)
plt.show()
```

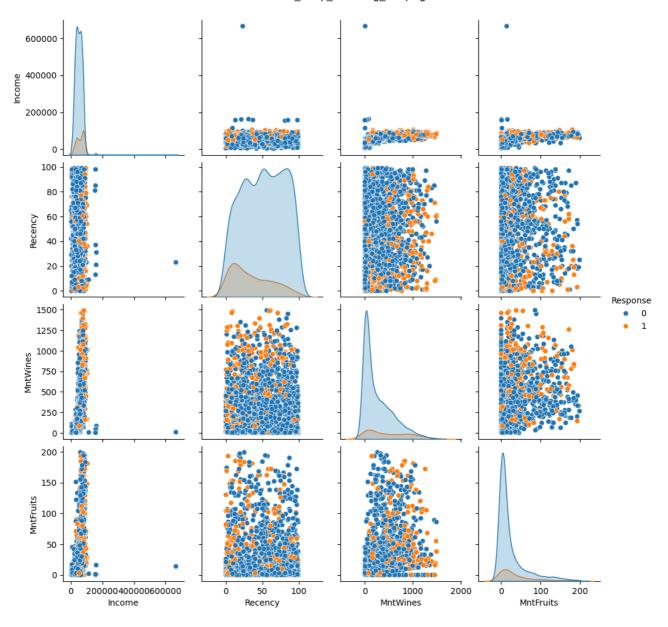
→ <ipython-input-48-b064766380e7>:2: FutureWarning: The default of observed=False is de income_response = df.groupby(income_bins)['Response'].mean() * 100



Pair Plot: Correlation between key variables

sns.pairplot(df[['Income', 'Recency', 'MntWines', 'MntFruits', 'Response']], hue='Respons plt.show()





Heatmap: Correlation Matrix

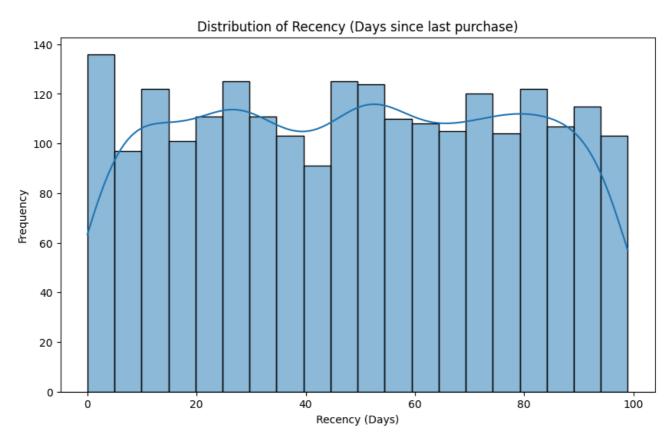
```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()
 \rightarrow
                                                                              Correlation Matrix of Numerical Features
                                                                                                                                                                                              1.00
                                      -0.0-20.0-20.0-20.0-20.0-4
                            Year_Birth -0.001.000.100.230.350.020.100.020.030.040.020.060.150.120.130.120.060.060.010.0-10.0-10.030
                                                                                                                                                        0.020.1-D.1-D.1-D.00
                               Income -0.010.111<mark>.00</mark>0.418.020.00.510.420.50.440.440.320.080.310.510.550.550.0180.330.270.090.03
idhome -0.000.220.441.000.04.010.580.370.440.340.370.310.220.310.50.50.50.50.450.010.160.240.170.080.04
                                                                                                                                                        0.13<mark>0.660.66</mark>0.44.31
-0.030.560.50.260.21
                              Kidhome -0.000.230.431.000.040.030.540.340.340.340.350.230.350.
                                                                                                                                                                                              0.75
                            Teenhome -0.00.35.020.04.00.020.00.180.260.200.160.020.390.160.100.050.130.040.040.140.140.020.00
                                                                                                                                                        -0.150.140.140.180.13
                              Recency -0.050.020.000.010.021.000.020.000.020.000.020.000.010.030.000.020.000.020.000.01
                                                                                                                                                        -0.200.020.020.040.01
                            MntWines -0.020.10.550.50.000.021.00.390.50.400.390.390.010.540.640.650.320.000.370.470.350.210.04
                                                                                                                                                         0.25<mark>0.890.85</mark>0.4B.51
                                                                                                                                                        0.130.610.61<mark>0.19</mark>.16
                             MntFruits -0.000.020.430.3-70.1-80.000.391.000.5-40.590.5 70.390.130.300.490.400.420.010.010.0220.190.0-0.01
                                                                                                                                                                                              0.50
                 MntMeatProducts -0.00.03.550.440.26.02.50.54.00.570.520.350.12.29.72.480.50.020.10.370.310.040.02
MntFishProducts -0.02.04.440.390.20.000.440.590.570.000.500.420.140.290.530.4400.490.000.020.200.260.000.02
MntSweetProducts -0.020.020.440.3370.160.020.350.570.570.520.000.370.120.350.490.450.420.000.030.260.240.030.02
                                                                                                                                                        0.24<mark>0.840.84</mark>0.25.31
                                                                                                                                                        0.11<mark>0.640.64</mark>0.15.18
                                                                                                                                                        0.120.600.66<mark>0.28</mark>.20
                        MntGoldProds -0.0-0.0 0.320.350.0 0.0 20.390.390.350.420.371.0 0.0 50.420.440.380.250.120.0 20.180.170.0 50.0 3
                                                                                                                                                        0.140.520.520.120.19
                                                                                                                                                                                             - 0.25
               NumDealsPurchases -0.040.060.080.220.390.000.010.130.120.140.120.051.000.230.010.070.350.020.020.140.120.040.00
                                                                                                                                                        0.000.070.070.220.12
                                                                                                                                                        0.150.520.520.08.19
                 NumWebPurchases -0.020.150.380.360.160.000.540.300.250.250.250.420.250.060.380.560.060.040.160.140.160.030.020
             NumCatalogPurchases -0.00.12.59.50.10.050.640.450.70.530.490.440.010.381.000.520.50.100.140.320.310.100.02
                                                                                                                                                         0.22<mark>0.780.78</mark>0.34.35
               NumStorePurchases -0.010.13.550.50.050.00<mark>.640.460.430.460.450.380.070.500.52<mark>1.00</mark>0.430.070.180.210.180.090.02
NumWebVisitsMonth -0.010.120.50.450.130.020.320.420.5-0.450.420.250.350.050.5-0.410.000.060.030.240.140.010.02</mark>
                                                                                                                                                        0.04<mark>0.670.67</mark>0.20.21
-0.000.540.50.240.17
                                                                                                                                                                                             - 0.00
                     AcceptedCmp3 -0.04.060.020.010.040.030.060.010.020.000.000.120.020.040.160.010.061.060.080.080.090.070.01
                                                                                                                                                        0.250.050.050.130.43
                     AcceptedCmp4 -0.030.06.180.16.040.020.370.010.100.020.030.020.020.160.140.180.030.080.000.310.250.290.03
                                                                                                                                                        0.180.250.250.33.6
                                                                                                                                                                                             - -0.25
                      AcceptedCmp5 -0.0 D.0 10.330.2-D.19.0 00.470.220.370.200.260.180.180.140.320.240.280.080.311.000.400.220.01
                                                                                                                                                         0.330.470.470.50.7
                      AcceptedCmp1 -0.020.0 D.270.1-0.1-0.020.350.190.3 D.260.240.170.120.160.3 D.180.190.090.2 50.401.00.180.03
                                                                                                                                                        0.290.380.380.5B.6
                      0.170.140.140.440.46
                             -0.040.040.040.020.02
                                                                                                                                                                                                -0.50
                      Z CostContact -
                           Z_Revenue -
                             Response -0.020.020.130.040.150.200.250.130.240.110.120.140.000.150.220.040.000.250.180.330.290.170.000
                                                                                                                                                         1.00<mark>0.270.270.31</mark>0.43
                       Total_Revenue -0.020.10.660.5-0.14.020.89.610.840.640.6(0.520.010.520.780.670.50.050.250.470.380.140.04

ROI -0.020.10.660.5-0.140.020.890.610.840.640.6(0.520.010.520.780.670.50.050.250.470.380.140.04
                                                                                                                                                         0.27<mark>1.00.00</mark>0.410.46
                                                                                                                                                                                               -0.75
                                                                                                                                                         0.27<mark>1.00.00</mark>0.40.46
                                                                                                                                                        -0.3<del>2</del>0.4<del>1</del>0.41<mark>1.060.9</mark>
0.430.460.46<mark>0.96</mark>.00
                                    CPA -0.020.1-D.440.200.180.040.430.140.250.1-50.230.120.220.080.340.200.240.130.330.5-D.5-0.440.02
                      Total_Accepted -0.040.000.310.2-10.1-10.010.510.160.310.180.200.190.110.190.350.210.110.420.610.720.640.460.02
                                                                                                                          AcceptedCmp4 -
AcceptedCmp5 -
                                                                                                                 NumWebVisitsMonth
                                                                                                                     AcceptedCmp3
                                                                                                                                                             Revenue
                                                                                                    NumWebPurchases
                                                                                                             NumStorePurchases
                                                                                                                                       AcceptedCmp2
                                                                                                                                                                           Accepted
                                                                              MntMeatProducts
                                                                                   MntFishProducts
                                                                                       MntSweetProducts
                                                                                            MntGoldProds
                                                                                                NumDealsPurchases
                                                                                                         NumCatalogPurchases
                                                                                                                                                              Total
                                                                                                                                                                           Total
```

Histogram: Distribution of Recency

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Recency'], bins=20, kde=True)
plt.title('Distribution of Recency (Days since last purchase)')
plt.xlabel('Recency (Days)')
plt.ylabel('Frequency')
plt.show()
```





Pie Chart: Response Rate

```
Suggested code may be subject to a license | MarkKostantine/Spotify-Visualization-ML-Prediction
plt.figure(figsize=(8, 8))
plt.pie(response_rate, labels=response_rate.index, autopct='%1.1f%%',
startangle=90)
plt.title('Overall Response Rate')
plt.show()
```



Overall Response Rate