+ Code + Text

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

```
df = pd.read_csv('campaign - campaign.csv')
```

```
df.head()
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumCatalogPurcha
0	1826	1970	Graduation	Divorced	$84,835.00	0	0	6/16/14	0	189	...	
1	1	1961	Graduation	Single	$57,091.00	0	0	6/15/14	0	464	...	
2	10476	1958	Graduation	Married	$67,267.00	0	1	5/13/14	0	134	...	
3	1386	1967	Graduation	Together	$32,474.00	1	1	5/11/14	0	10	...	
4	5371	1989	Graduation	Single	$21,474.00	1	0	4/8/14	0	6	...	

5 rows × 27 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2239 entries, 0 to 2238
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   2239 non-null   int64 
 1   Year_Birth           2239 non-null   int64 
 2   Education            2239 non-null   object
 3   Marital_Status       2239 non-null   object
 4   Income               2239 non-null   object
 5   Kidhome              2239 non-null   int64 
 6   Teenhome             2239 non-null   int64 
 7   Dt_Customer          2239 non-null   object
 8   Recency              2239 non-null   int64 
 9   MntWines             2239 non-null   int64 
 10  MntFruits            2239 non-null   int64 
 11  MntMeatProducts      2239 non-null   int64 
 12  MntFishProducts      2239 non-null   int64 
 13  MntSweetProducts     2239 non-null   int64 
 14  MntGoldProds         2239 non-null   int64 
 15  NumDealsPurchases    2239 non-null   int64 
 16  NumWebPurchases      2239 non-null   int64 
 17  NumCatalogPurchases  2239 non-null   int64 
 18  NumStorePurchases    2239 non-null   int64 
 19  NumWebVisitsMonth    2239 non-null   int64 
 20  AcceptedCmp3         2239 non-null   int64 
 21  AcceptedCmp4         2239 non-null   int64 
 22  AcceptedCmp5         2239 non-null   int64 
 23  AcceptedCmp1         2239 non-null   int64 
 24  AcceptedCmp2         2239 non-null   int64 
 25  Complain             2239 non-null   int64 
 26  Country              2239 non-null   object
dtypes: int64(22), object(5)
memory usage: 472.4+ KB
```

```
df.isna().sum()
```

	0
ID	0
Year_Birth	0
Education	0
Marital_Status	0
Income	0
Kidhome	0
Teenhome	0
Dt_Customer	0
Recency	0
MntWines	0
MntFruits	0
MntMeatProducts	0
MntFishProducts	0
MntSweetProducts	0
MntGoldProds	0
NumDealsPurchases	0
NumWebPurchases	0
NumCatalogPurchases	0
NumStorePurchases	0
NumWebVisitsMonth	0
AcceptedCmp3	0
AcceptedCmp4	0
AcceptedCmp5	0
AcceptedCmp1	0
AcceptedCmp2	0
Complain	0
Country	0

dtype: int64

```
df.describe()
```

	ID	Year_Birth	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	M
count	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	
mean	5590.444841	1968.802144	0.443948	0.506476	49.121036	304.067441	26.307727	167.016525	37.538633	
std	3246.372471	11.985494	0.538390	0.544555	28.963662	336.614830	39.781468	225.743829	54.637617	
min	0.000000	1893.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2827.500000	1959.000000	0.000000	0.000000	24.000000	24.000000	1.000000	16.000000	3.000000	
50%	5455.000000	1970.000000	0.000000	0.000000	49.000000	174.000000	8.000000	67.000000	12.000000	
75%	8423.500000	1977.000000	1.000000	1.000000	74.000000	504.500000	33.000000	232.000000	50.000000	
max	11191.000000	1996.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	

8 rows × 22 columns

```
numerical_features = ['Year_Birth', 'Recency', 'MntWines', 'MntFruits',
                      'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
                      'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
                      'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth']
def count_outliers(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
```

```
    # Count outliers
    return ((series < lower_bound) | (series > upper_bound)).sum()

# Count outliers in each numerical column
outliers_count = {feature: count_outliers(df[feature]) for feature in numerical_features}

# Display the count of outliers
outliers_count_df = pd.DataFrame.from_dict(outliers_count, orient='index', columns=['Outliers Count'])
print(outliers_count_df)
```

```
                     Outliers Count
Year_Birth                        3
Recency                           0
MntWines                         35
MntFruits                       227
MntMeatProducts                 175
MntFishProducts                 223
MntSweetProducts                248
MntGoldProds                    207
NumDealsPurchases                86
NumWebPurchases                   4
NumCatalogPurchases              23
NumStorePurchases                 0
NumWebVisitsMonth                 8
```

```
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filter out outliers
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Remove outliers from each numerical column
for feature in numerical_features:
    df = remove_outliers(df, feature)


# ANOVA Test
# Group data by 'Education' and compute the mean income for each group
import scipy.stats as stats
# ANOVA Test
# Group data by 'Education' and compute the mean income for each group
import scipy.stats as stats

# Convert 'Income' column to numeric, removing '$' and ','
df['Income'] = df['Income'].str.replace('$', '').str.replace(',', '').astype(float)

education_groups = df.groupby('Education')['Income'].apply(list)

# Perform ANOVA test
f_statistic, p_value = stats.f_oneway(*education_groups)

# Display the results
print(f"F-Statistic: {f_statistic}")
print(f"P-Value: {p_value}")

# Interpretation of results
alpha = 0.05  # Significance level
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in income across education levels.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in income across education levels.")

# Optional: Visualize the income distribution by education level
plt.figure(figsize=(12, 8))
sns.boxplot(x='Education', y='Income', data=df, palette='Set2')
plt.title('Income Distribution by Education Level')
plt.xticks(rotation=45)
education_groups = df.groupby('Education')['Income'].apply(list)

# Perform ANOVA test
f_statistic, p_value = stats.f_oneway(*education_groups)

# Display the results
print(f"F-Statistic: {f_statistic}")
print(f"P-Value: {p_value}")

# Interpretation of results
alpha = 0.05  # Significance level
if p_value < alpha:
```

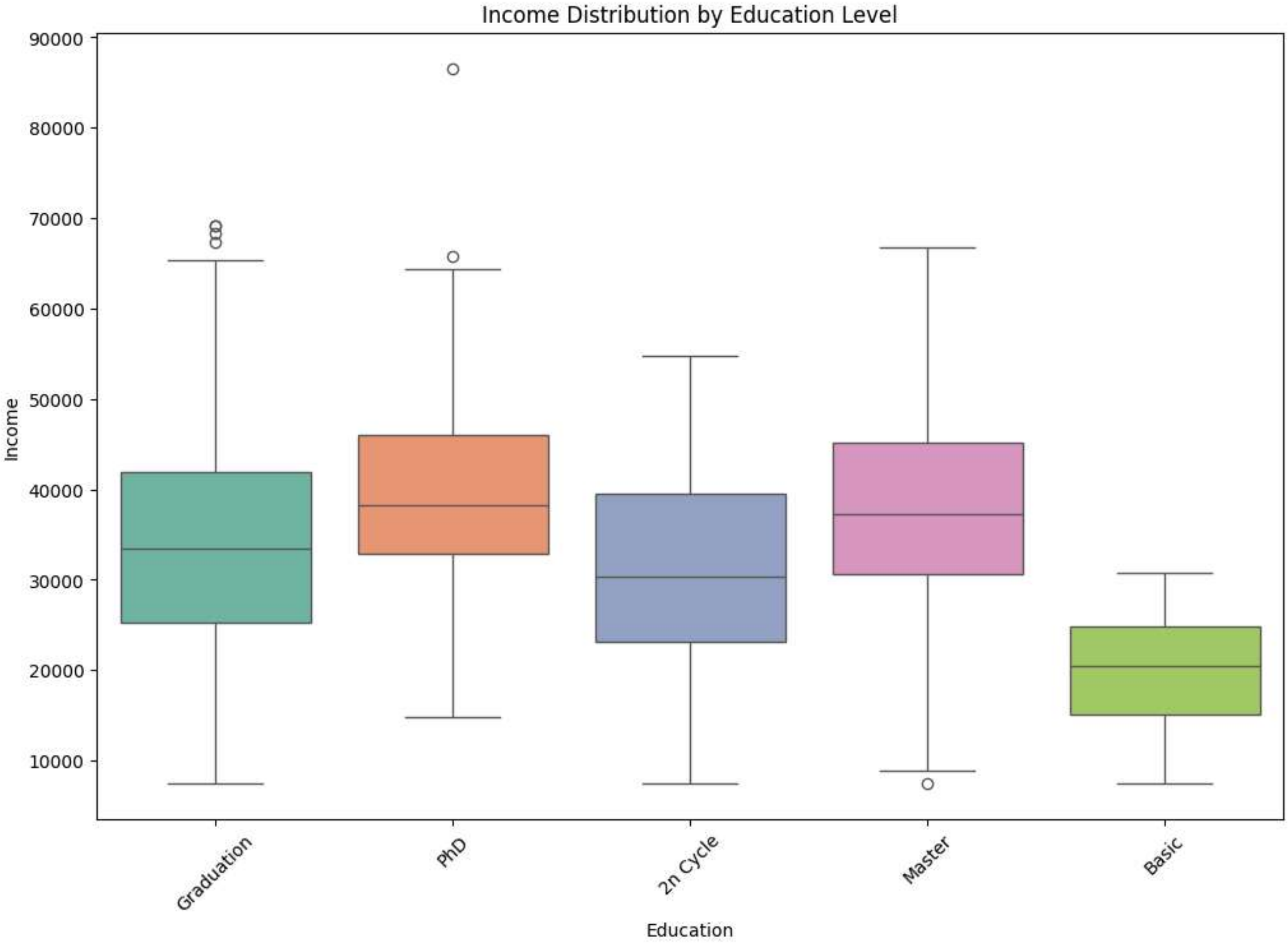
```
    print("Reject the null hypothesis: There is a significant difference in income across education levels.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in income across education levels.")

# Optional: Visualize the income distribution by education level
plt.figure(figsize=(12, 8))
sns.boxplot(x='Education', y='Income', data=df, palette='Set2')
plt.title('Income Distribution by Education Level')
plt.xticks(rotation=45)
```

```
F-Statistic: nan
P-Value: nan
Fail to reject the null hypothesis: There is no significant difference in income across education levels.
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-b17759d7e05c> in <cell line: 28>()
     26
     27 # Optional: Visualize the income distribution by education level
---> 28 plt.figure(figsize=(12, 8))
     29 sns.boxplot(x='Education', y='Income', data=df, palette='Set2')
     30 plt.title('Income Distribution by Education Level')

TypeError: 'module' object is not callable
```

Next steps: Explain error

```
# ANOVA Test
# Group data by 'Education' and compute the mean income for each group
import scipy.stats as stats
import matplotlib.pyplot as plt # Ensure matplotlib.pyplot is imported correctly

# Convert 'Income' to numerical if it's not already
df['Income'] = pd.to_numeric(df['Income'], errors='coerce')

education_groups = df.groupby('Education')['Income'].apply(list)

# Perform ANOVA test
f_statistic, p_value = stats.f_oneway(*education_groups)

# Display the results
print(f"F-Statistic: {f_statistic}")
print(f"P-Value: {p_value}")

# Interpretation of results
alpha = 0.05  # Significance level
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in income across education levels.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in income across education levels.")

# Optional: Visualize the income distribution by education level
plt.figure(figsize=(12, 8)) # Call the figure function from the plt module
import seaborn as sns # Import seaborn for boxplot
sns.boxplot(x='Education', y='Income', data=df, palette='Set2')
plt.title('Income Distribution by Education Level')
plt.xticks(rotation=45)
```

```
F-Statistic: nan
P-Value: nan
Fail to reject the null hypothesis: There is no significant difference in income across education levels.
<ipython-input-22-465985bba4bc>:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  sns.boxplot(x='Education', y='Income', data=df, palette='Set2')
([0, 1, 2, 3, 4],
 [Text(0, 0, 'Graduation'),
  Text(1, 0, 'PhD'),
  Text(2, 0, '2n Cycle'),
  Text(3, 0, 'Master'),
  Text(4, 0, 'Basic')])
```




```
# Calculate total spending by summing up all spending columns
df['Total_Spending'] = df[['MntWines', 'MntFruits', 'MntMeatProducts',
                           'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']].sum(axis=1)

# Check the first few rows to verify
print(df[['Income', 'Total_Spending']].head())
```

```
     Income  Total_Spending
2   67267.0             251
3   32474.0              11
4   21474.0              91
7   44931.0              96
13  26872.0              72
```

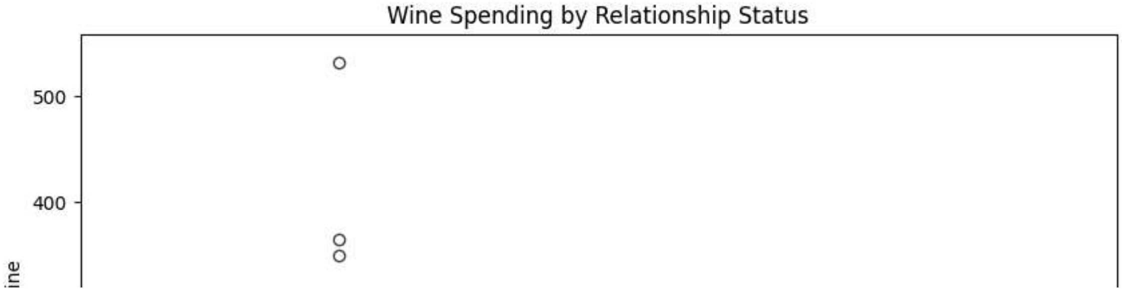
```
import scipy.stats as stats
df['Income'] = pd.to_numeric(df['Income'], errors='coerce')
df['Total_Spending'] = pd.to_numeric(df['Total_Spending'], errors='coerce')
df = df.dropna(subset=['Income', 'Total_Spending'])

# Calculate Pearson correlation coefficient
correlation, p_value = stats.pearsonr(df['Income'], df['Total_Spending'])

print(f"Pearson Correlation Coefficient: {correlation}")
print(f"P-Value: {p_value}")
```

```
# Interpretation
alpha = 0.05
if p_value < alpha:
    print("There is a significant correlation between income and total spending.")
else:
    print("There is no significant correlation between income and total spending.")
```

```
Pearson Correlation Coefficient: 0.4726446220666236
P-Value: 1.6010711861649903e-55
There is a significant correlation between income and total spending.
```

```
# Map marital status to 'In couple' or 'Alone'
df['Relationship_Status'] = df['Marital_Status'].map({
    'Married': 'In couple',
    'Together': 'In couple',
    'Divorced': 'Alone',
    'Single': 'Alone',
    'Absurd': 'Alone',
    'Widow': 'Alone',
    'YOLO': 'Alone'
})

# Check the mapping
print(df[['Marital_Status', 'Relationship_Status']].drop_duplicates())

# Group data by relationship status and calculate mean spending on wine
grouped_data = df.groupby('Relationship_Status')['MntWines']

# Perform independent samples t-test
group_in_couple = grouped_data.get_group('In couple')
group_alone = grouped_data.get_group('Alone')

# Ensure both groups have data
if not group_in_couple.empty and not group_alone.empty:
    t_statistic, p_value = stats.ttest_ind(group_in_couple, group_alone, equal_var=False)

    # Display results
    print(f"T-Statistic: {t_statistic}")
    print(f"P-Value: {p_value}")

    # Interpretation
    alpha = 0.05  # Significance level
    if p_value < alpha:
        print("Reject the null hypothesis: There is a significant difference in wine spending between couples and individuals living al
    else:
        print("Fail to reject the null hypothesis: There is no significant difference in wine spending between couples and individuals
else:
    print("One or both groups are empty. Check the data and try again.")

# Optional: Visualize the wine spending by relationship status
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.boxplot(x='Relationship_Status', y='MntWines', data=df)
plt.title('Wine Spending by Relationship Status')
plt.xlabel('Relationship Status')
plt.ylabel('Spending on Wine')
plt.show()
```

```
    Marital_Status Relationship_Status
2          Married          In couple
3         Together          In couple
4           Single              Alone
17        Divorced              Alone
28           Widow              Alone
286          Alone                NaN
T-Statistic: 0.6346714757271168
P-Value: 0.525825023463167
Fail to reject the null hypothesis: There is no significant difference in wine spending between couples and individuals living alon
```




```
# Calculate the median income
median_income = df['Income'].median()

# Create income brackets
df['Income_Bracket'] = np.where(df['Income'] < median_income, 'Below Median', 'Above Median')

# Create a column indicating if any campaign was accepted
df['Accepted_Any_Campaign'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1) > 0

# Group by income bracket and calculate acceptance rates
acceptance_rates = df.groupby('Income_Bracket')['Accepted_Any_Campaign'].mean()

# Print acceptance rates
print(acceptance_rates)

# Perform a Chi-Square Test for independence
contingency_table = pd.crosstab(df['Income_Bracket'], df['Accepted_Any_Campaign'])
chi2_stat, p_value, _, _ = stats.chi2_contingency(contingency_table)

# Display results
print(f"Chi-Square Statistic: {chi2_stat}")
print(f"P-Value: {p_value}")

# Interpretation
alpha = 0.05  # Significance level
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in campaign acceptance between income brackets.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in campaign acceptance between income brackets.")
```

```
Income_Bracket
Above Median    0.075510
Below Median    0.078029
Name: Accepted_Any_Campaign, dtype: float64
Chi-Square Statistic: 0.0007659611921822663
P-Value: 0.9779205624803663
Fail to reject the null hypothesis: There is no significant difference in campaign acceptance between income brackets.
```

Additional Hypothesis Tests performed

```
# Correlation analysis for spending on wine vs. number of children
correlation, p_value = stats.pearsonr(df['Kidhome'], df['MntWines'])

print(f"Pearson Correlation Coefficient: {correlation}")
print(f"P-Value: {p_value}")
```