In [1]:
```python
# Import the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Import seaborn
import seaborn as sns

# Apply the default theme
sns.set_theme()
%matplotlib inline
```

In [2]:
```python
data = pd.read_csv("Raw Data.csv")
data.head() # to check the first five rows
```

Out[2]:

	Campaign ID	Campaign Name	Audience	Age	Geography	Reach	Impressions	Frequency	Clicks	Unique Clicks	Unique Link Clicks (ULC)	Click-Through Rate (CTR)	Unique Click-Through Rate (Unique CTR)	Amount Spent in INR	Cost Per Click (CPC)	Cost per Result (CPR)
0	Campaign 1	SHU_6 (Educators and Principals)	Educators and Principals	25-34	Group 1 (Australia, Canada, United Kingdom, Gh...	11387	23283	2.044700	487	406	180	2.091655	3.565469	1092.24	2.242790	6.07
1	Campaign 1	SHU_6 (Educators and Principals)	Educators and Principals	35-44	Group 1 (Australia, Canada, United Kingdom, Gh...	8761	15683	1.790092	484	376	154	3.086144	4.291748	835.46	1.726165	5.43
2	Campaign 1	SHU_6 (Educators and Principals)	Educators and Principals	45-54	Group 1 (Australia, Canada, United Kingdom, Gh...	2867	6283	2.191489	198	145	65	3.151361	5.057551	319.38	1.613038	4.91
3	Campaign 1	SHU_6 (Educators and Principals)	Educators and Principals	55-64	Group 1 (Australia, Canada, United Kingdom, Gh...	889	1890	2.125984	49	40	21	2.592593	4.499438	86.25	1.760117	4.11
4	Campaign 2	SHU3_ (Students Apart from India and US)	Students	18-24	Group 2 (Australia, Canada, United Kingdom, Gh...	29675	39161	1.319663	2593	1994	1095	6.621384	6.719461	1193.94	0.460448	1.09

In [3]:
```python
# to get the shape of the dataset
data.shape
```

Out[3]: (33, 16)

In [4]:
```python
data.info() # to get the Structure from dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 16 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Campaign ID                             33 non-null     object
 1   Campaign Name                           33 non-null     object
 2   Audience                                33 non-null     object
 3   Age                                     33 non-null     object
 4   Geography                               33 non-null     object
 5   Reach                                   33 non-null     int64
 6   Impressions                             33 non-null     int64
 7   Frequency                               33 non-null     float64
 8   Clicks                                  33 non-null     int64
 9   Unique Clicks                           33 non-null     int64
 10  Unique Link Clicks (ULC)                33 non-null     int64
 11  Click-Through Rate (CTR)                33 non-null     float64
 12  Unique Click-Through Rate (Unique CTR)  33 non-null     float64
 13  Amount Spent in INR                     33 non-null     float64
 14  Cost Per Click (CPC)                    33 non-null     float64
 15  Cost per Result (CPR)                   33 non-null     float64
dtypes: float64(6), int64(5), object(5)
memory usage: 4.2+ KB
```

In [5]:
```python
# to get the statistical data
data.describe()
```

Out[5]:

	Reach	Impressions	Frequency	Clicks	Unique Clicks	Unique Link Clicks (ULC)	Click-Through Rate (CTR)	Unique Click-Through Rate (Unique CTR)	Amount Spent in INR	Cost Per Click (CPC)	Cost per Result (CPR)
count	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000	33.000000
mean	5723.272727	8783.636364	1.418234	364.393939	288.000000	159.303030	4.814551	5.143724	366.321515	3.176343	7.640606
std	7671.821261	11327.981290	0.471318	556.340326	432.381125	257.960933	2.901685	2.481833	298.627967	2.841653	7.813851
min	91.000000	103.000000	1.042614	9.000000	8.000000	3.000000	1.668892	2.026599	47.260000	0.290938	0.690000
25%	889.000000	1874.000000	1.131868	49.000000	44.000000	27.000000	2.592593	3.605150	117.900000	0.683483	1.640000
50%	2557.000000	3146.000000	1.174759	135.000000	111.000000	63.000000	3.982684	4.416037	283.170000	1.760117	5.430000
75%	6145.000000	12372.000000	1.665733	325.000000	246.000000	129.000000	6.621384	6.534091	487.520000	5.486358	11.110000
max	30110.000000	39161.000000	3.169081	2593.000000	1994.000000	1095.000000	12.951807	12.131148	1193.940000	10.184692	30.550000

In [6]:
```python
#To know more about the dataset with transpose - here T is for the transpose

data.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Reach	33.0	5723.272727	7671.821261	91.000000	889.000000	2557.000000	6145.000000	30110.000000
Impressions	33.0	8783.636364	11327.981290	103.000000	1874.000000	3146.000000	12372.000000	39161.000000
Frequency	33.0	1.418234	0.471318	1.042614	1.131868	1.174759	1.665733	3.169081
Clicks	33.0	364.393939	556.340326	9.000000	49.000000	135.000000	325.000000	2593.000000
Unique Clicks	33.0	288.000000	432.381125	8.000000	44.000000	111.000000	246.000000	1994.000000
Unique Link Clicks (ULC)	33.0	159.303030	257.960933	3.000000	27.000000	63.000000	129.000000	1095.000000
Click-Through Rate (CTR)	33.0	4.814551	2.901685	1.668892	2.592593	3.982684	6.621384	12.951807
Unique Click-Through Rate (Unique CTR)	33.0	5.143724	2.481833	2.026599	3.605150	4.416037	6.534091	12.131148
Amount Spent in INR	33.0	366.321515	298.627967	47.260000	117.900000	283.170000	487.520000	1193.940000
Cost Per Click (CPC)	33.0	3.176343	2.841653	0.290938	0.683483	1.760117	5.486358	10.184692
Cost per Result (CPR)	33.0	7.640606	7.813851	0.690000	1.640000	5.430000	11.110000	30.550000

Data Cleaning and filling missing values

In [7]:
```python
data.apply(lambda x: sum(x.isnull()),axis=0) # checking missing values in each column of train dataset
```

Out[7]:
```
Campaign ID                               0
Campaign Name                             0
Audience                                  0
Age                                       0
Geography                                 0
Reach                                     0
Impressions                               0
Frequency                                 0
Clicks                                    0
Unique Clicks                             0
Unique Link Clicks (ULC)                  0
Click-Through Rate (CTR)                  0
Unique Click-Through Rate (Unique CTR)    0
Amount Spent in INR                       0
Cost Per Click (CPC)                      0
Cost per Result (CPR)                     0
dtype: int64
```

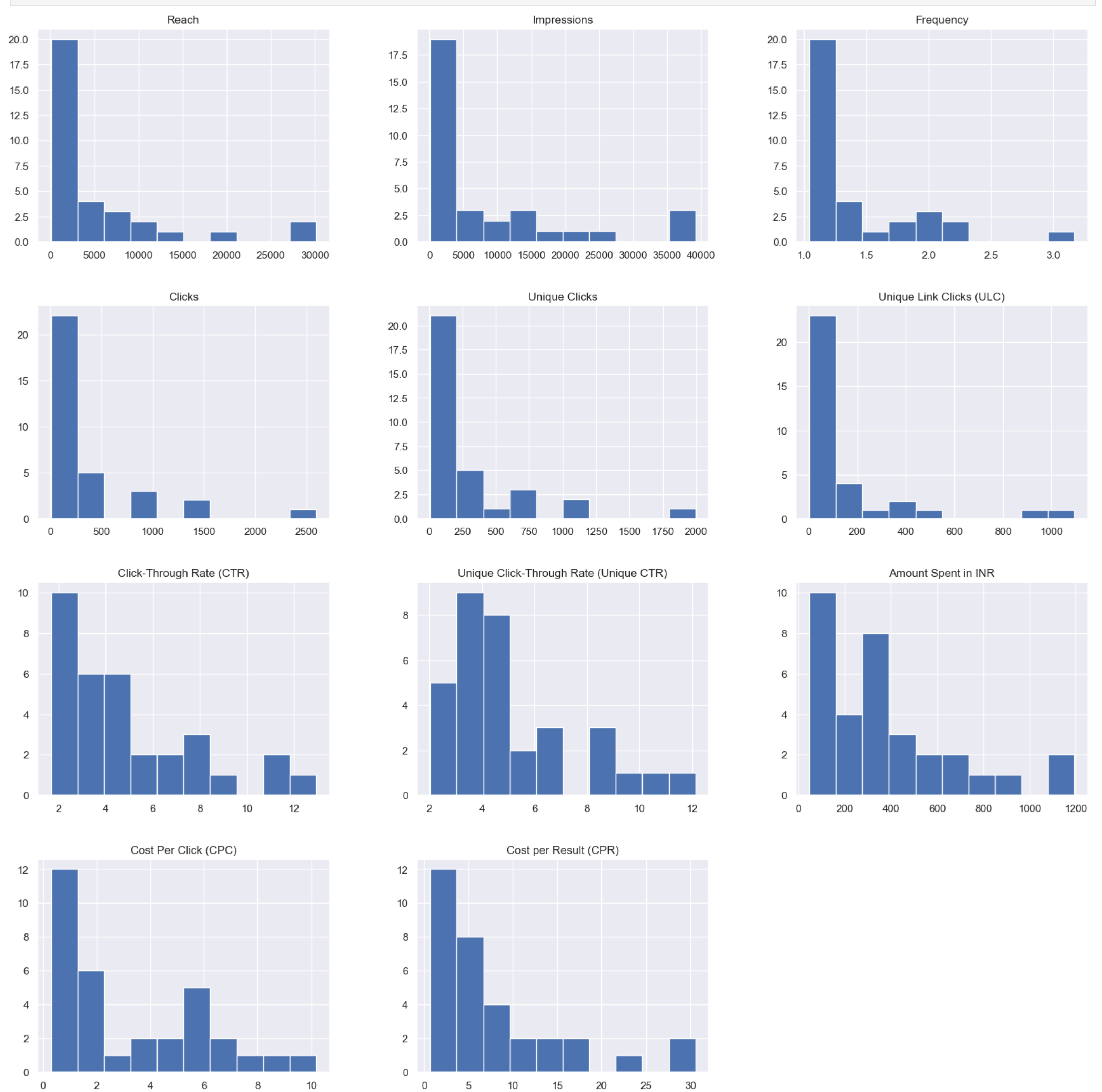
In [8]:
```python
#Now let’s check that if our dataset have null values or not
data.isnull().head(10)
```

Out[8]:

	Campaign ID	Campaign Name	Audience	Age	Geography	Reach	Impressions	Frequency	Clicks	Unique Clicks	Unique Link Clicks (ULC)	Click-Through Rate (CTR)	Unique Click-Through Rate (Unique CTR)	Amount Spent in INR	Cost Per Click (CPC)	Cost per Result (CPR)
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

Data Visualization

In [9]:
```python
#Plotting the data distribution plots

p = data.hist(figsize = (20,20))
```

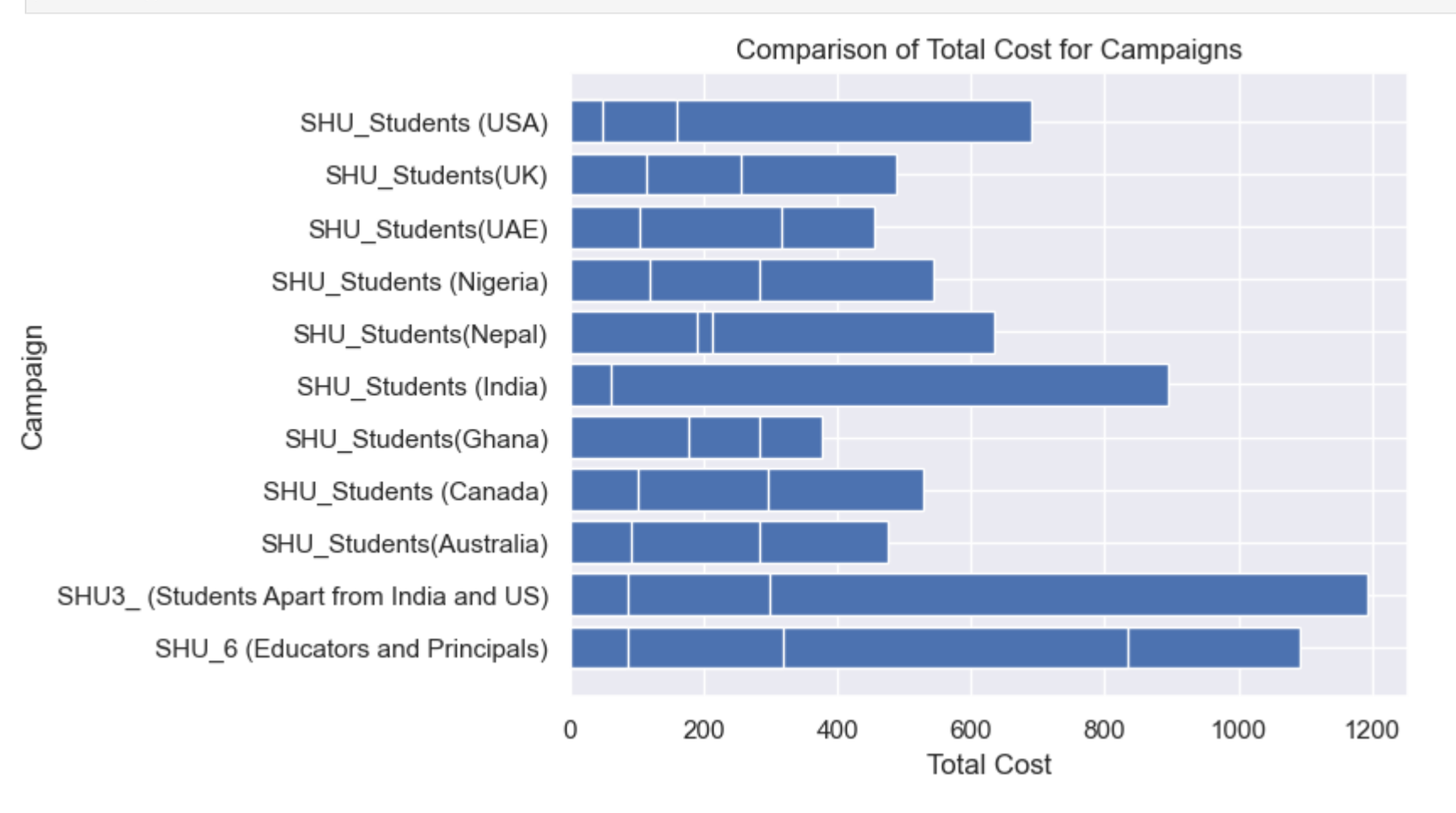


Correlation between all the features

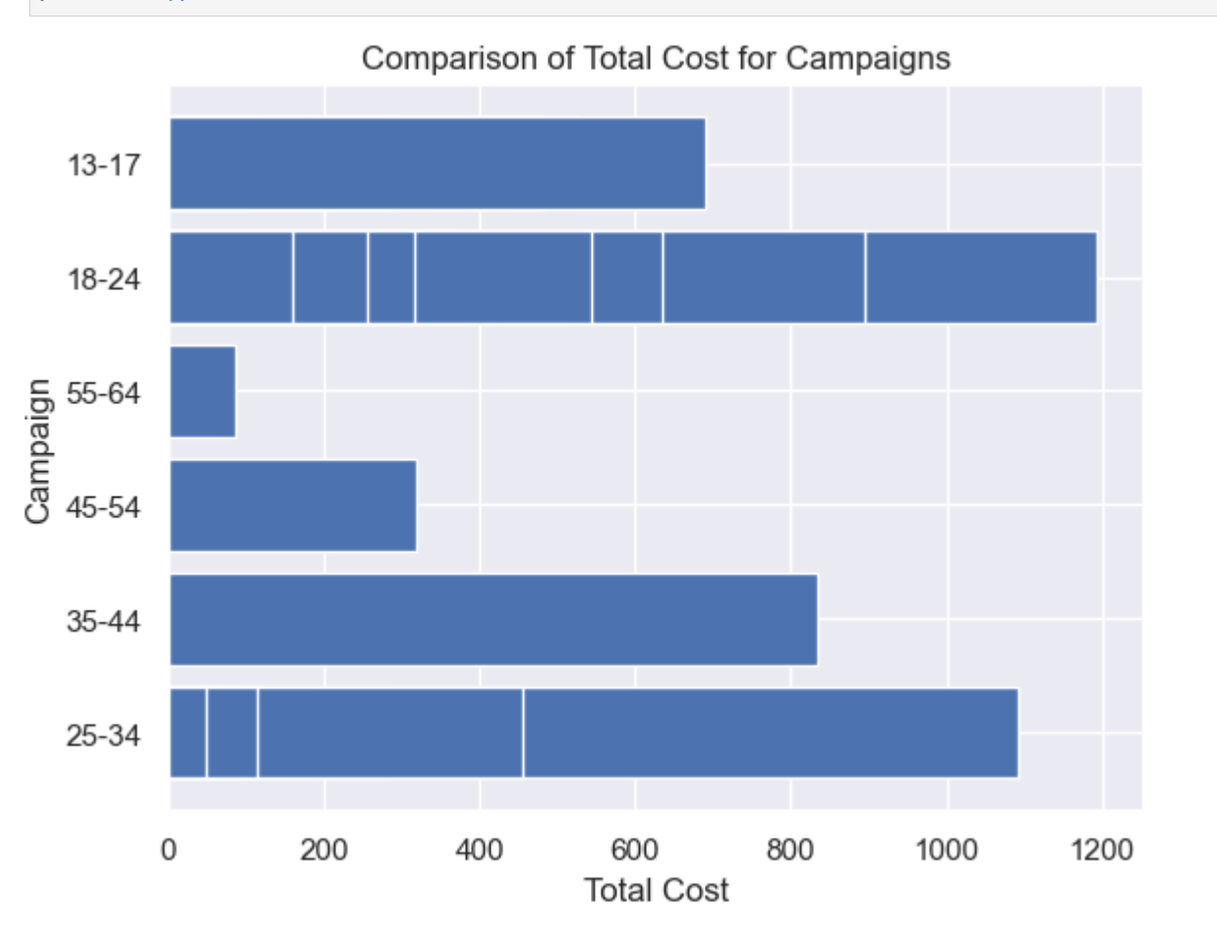
In [10]:
```python
#Correlation between all the features

plt.figure(figsize=(20,10))
# seaborn has an easy method to showcase heatmap
p = sns.heatmap(data.corr(), annot=True,cmap ='RdYlGn')
```

```
C:\Users\ELCOT\AppData\Local\Temp\ipykernel_14684\3144283168.py:5: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  p = sns.heatmap(data.corr(), annot=True,cmap ='RdYlGn')
```

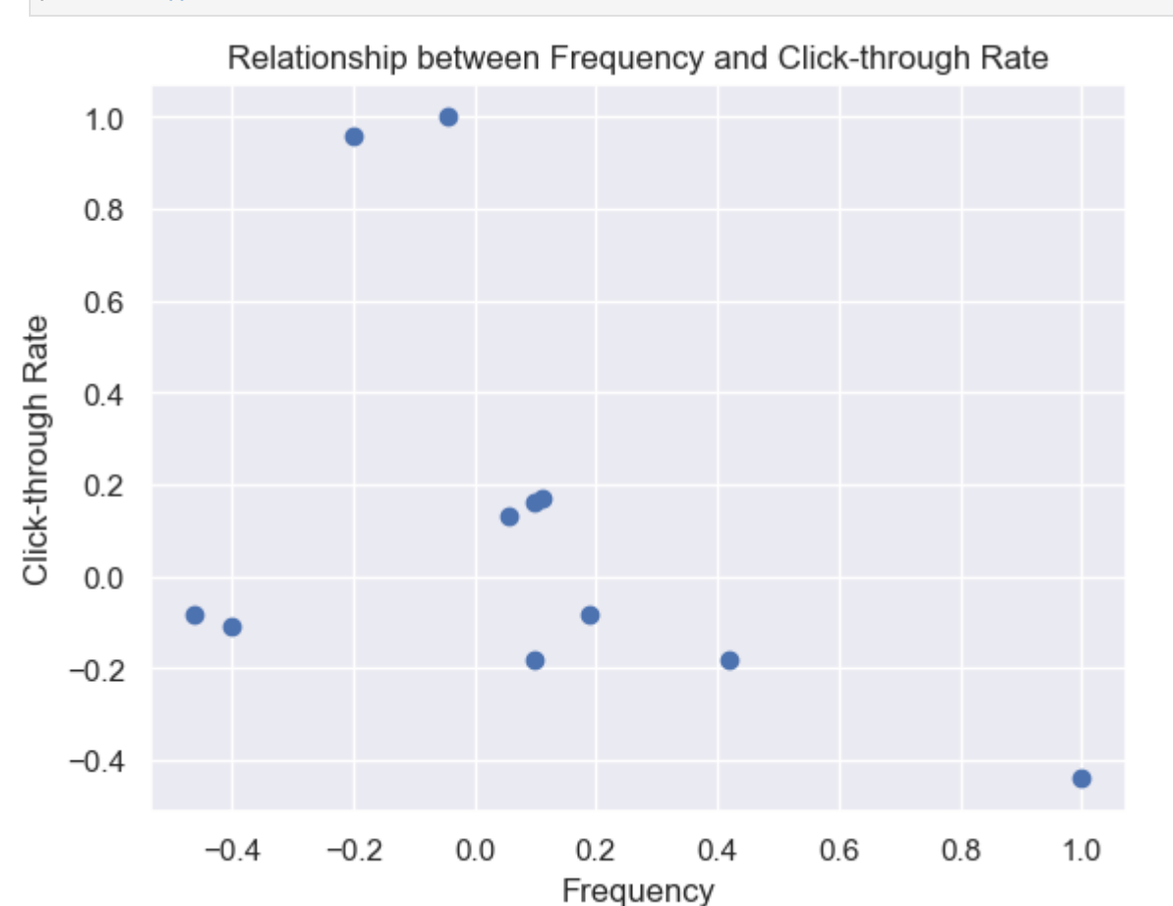
In [11]:
```python
# Extract the necessary columns from the data
campaigns = data['Campaign Name']
total_cost = data['Amount Spent in INR']

# Create a horizontal bar chart
plt.barh(campaigns, total_cost)

# Add labels and title
plt.xlabel('Total Cost')
plt.ylabel('Campaign')
plt.title('Comparison of Total Cost for Campaigns')

# Show the plot
plt.show()
```



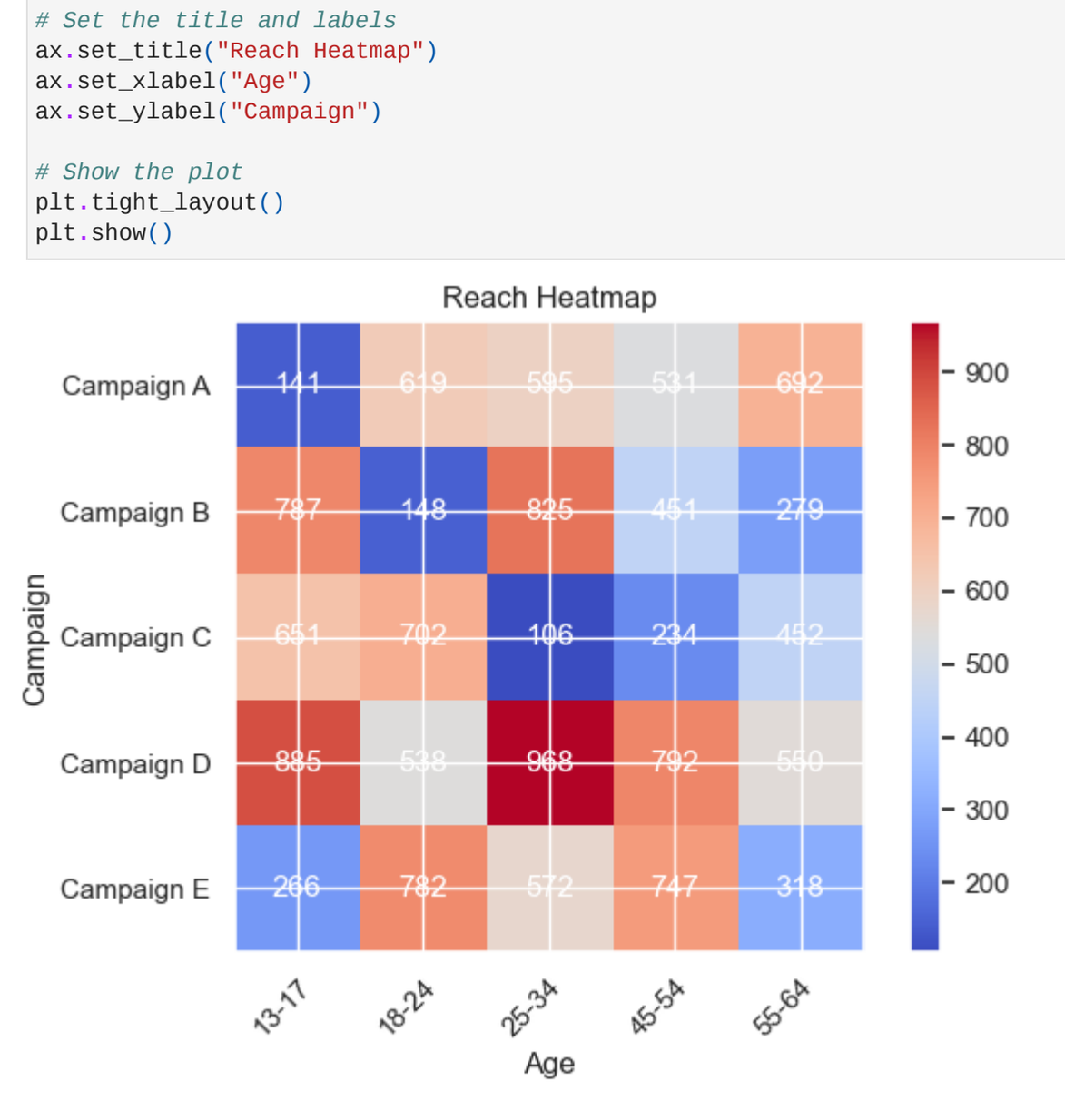
In [12]:
```python
# Extract the necessary columns from the data
campaigns = data['Age']
total_cost = data['Amount Spent in INR']

# Create a horizontal bar chart
plt.barh(campaigns, total_cost)

# Add labels and title
plt.xlabel('Total Cost')
plt.ylabel('Campaign')
plt.title('Comparison of Total Cost for Campaigns')

# Show the plot
plt.show()
```



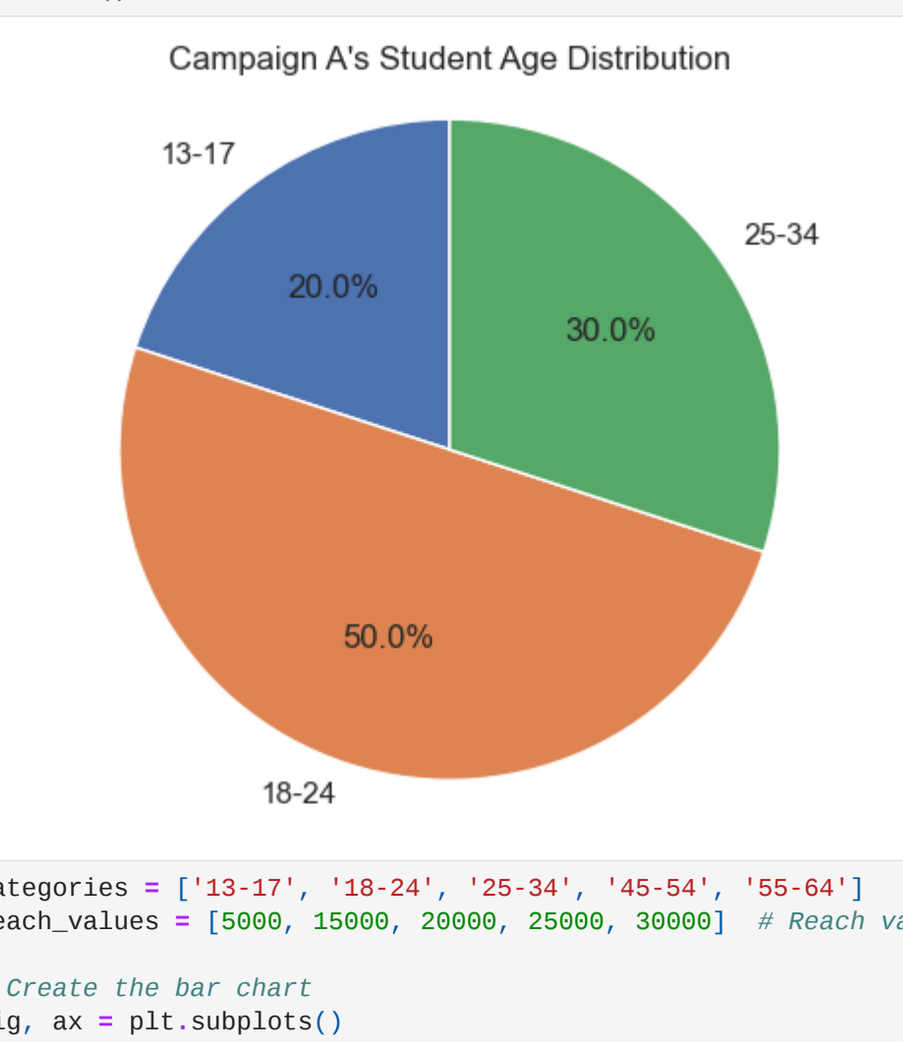
In [13]:
```python
# Sample data
frequency = [0.19,0.42,1.0,0.11,0.1,0.056,-0.044,-0.2,0.1,-0.46,-0.4]
click_through_rate = [-0.084,-0.18,-0.44,0.17,0.16,0.13,1.0,0.96,-0.18,-0.084,-0.11]

# Create scatter plot
plt.scatter(frequency, click_through_rate)

# Set labels and title
plt.xlabel('Frequency')
plt.ylabel('Click-through Rate')
plt.title('Relationship between Frequency and Click-through Rate')

# Display the plot
plt.show()
```



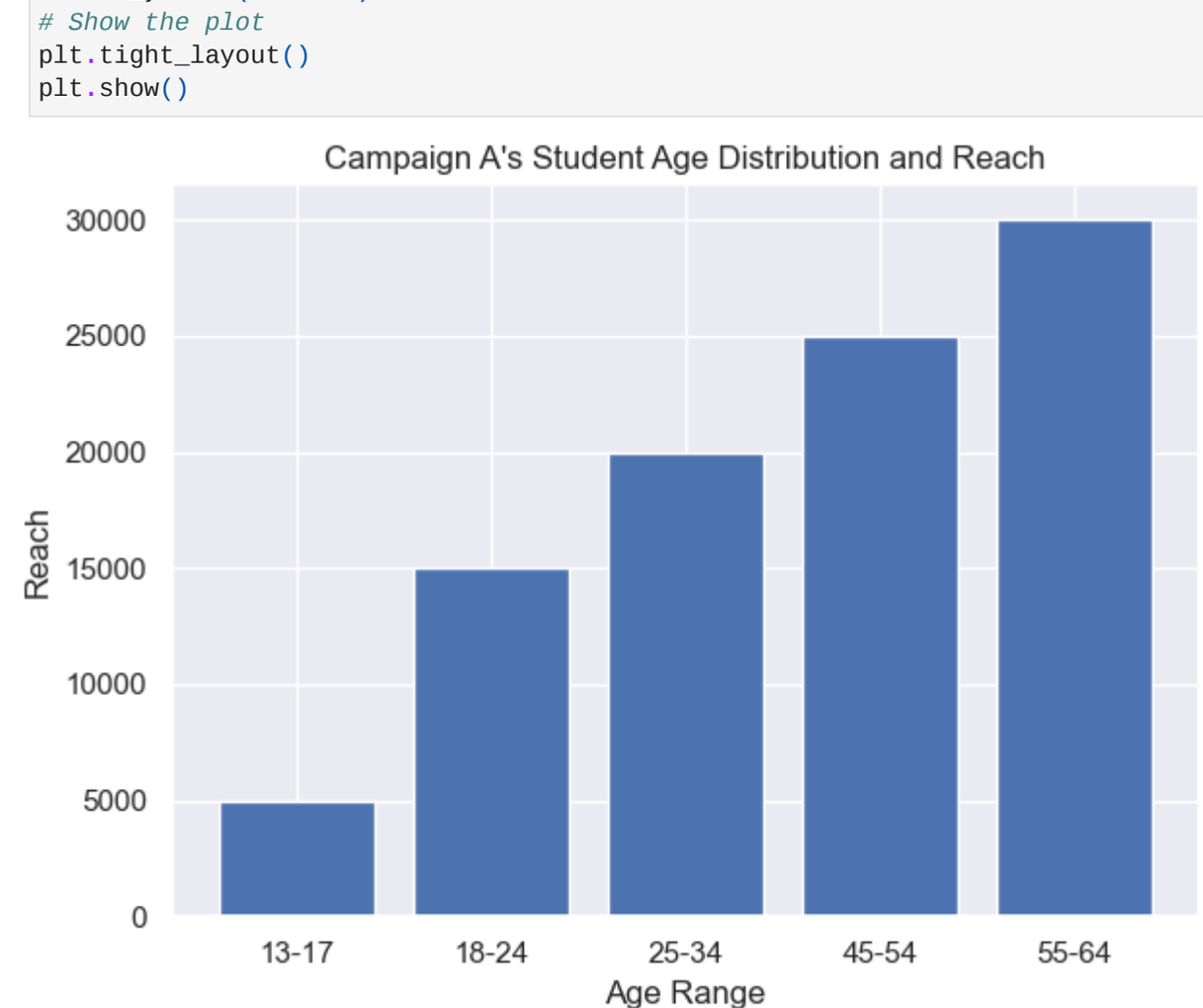
In [14]:
```python
campaigns = ['Campaign A', 'Campaign B', 'Campaign C','Campaign D','Campaign E']
ages = ['13-17', '18-24', '25-34','45-54','55-64']
reach_values = np.random.randint(100, 1000, size=(len(campaigns), len(ages)))

# Create a figure and axes
fig, ax = plt.subplots()

# Create the heatmap table
heatmap = ax.imshow(reach_values, cmap='coolwarm')

# Add colorbar
cbar = plt.colorbar(heatmap)

# Set the tick labels and positions for the x-axis (ages)
ax.set_xticks(np.arange(len(ages)))
ax.set_xticklabels(ages)

# Set the tick labels and positions for the y-axis (campaigns)
ax.set_yticks(np.arange(len(campaigns)))
ax.set_yticklabels(campaigns)

# Rotate the x-axis labels for better readability
plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")

# Loop over data dimensions and create text annotations in each cell
for i in range(len(campaigns)):
    for j in range(len(ages)):
        ax.text(j, i, reach_values[i, j], ha="center", va="center", color="w")

# Set the title and labels
ax.set_title("Reach Heatmap")
ax.set_xlabel("Age")
ax.set_ylabel("Campaign")

# Show the plot
plt.tight_layout()
plt.show()
```



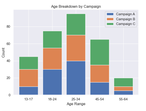
In [15]:
```python
categories = ['13-17', '18-24', '25-34']
composition = [20, 50, 30]  # Data should be sorted in descending order

# Create the pie chart
fig, ax = plt.subplots()
ax.pie(composition, labels=categories, autopct='%1.1f%%', startangle=90)

# Set the title
ax.set_title("Campaign A's Student Age Distribution")

# Equal aspect ratio ensures that pie is drawn as a circle
ax.axis('equal')

# Show the plot
plt.show()
```



In [16]:
```python
categories = ['13-17', '18-24', '25-34', '45-54', '55-64']
reach_values = [5000, 15000, 20000, 25000, 30000]  # Reach values for each age range

# Create the bar chart
fig, ax = plt.subplots()
x = np.arange(len(categories))
ax.bar(x, reach_values, align='center')

# Set the tick labels and positions
ax.set_xticks(x)
ax.set_xticklabels(categories)

# Set the title and labels
ax.set_title("Campaign A's Student Age Distribution and Reach")
ax.set_xlabel("Age Range")
ax.set_ylabel("Reach")
# Show the plot
plt.tight_layout()
plt.show()
```



In [17]:
```python
campaigns = ['Campaign A', 'Campaign B', 'Campaign C']
ages = ['13-17', '18-24', '25-34', '45-54', '55-64']

# Student age distribution for each campaign
campaign_a = [10, 30, 40, 15, 5]
campaign_b = [20, 25, 30, 20, 5]
campaign_c = [15, 20, 25, 30, 10]

# Create the stacked bar chart
fig, ax = plt.subplots()
x = np.arange(len(ages))

# Plot each campaign's age distribution as a stacked bar
ax.bar(x, campaign_a, label='Campaign A')
ax.bar(x, campaign_b, bottom=campaign_a, label='Campaign B')
ax.bar(x, campaign_c, bottom=np.array(campaign_a) + np.array(campaign_b), label='Campaign C')

# Set the tick labels and positions
ax.set_xticks(x)
ax.set_xticklabels(ages)

# Set the title and labels
ax.set_title("Age Breakdown by Campaign")
ax.set_xlabel("Age Range")
ax.set_ylabel("Count")

# Add a legend
ax.legend()

# Show the plot
plt.tight_layout()
plt.show()
```


Age Breakdown by Campaign
Campaign A
Campaign B
Campaign C
Count
0
20
40
60
13-17
18-24
25-34
35-44
45-54
55-64
Age Range

```
>> 1 1
>> 1 1
>> 1 1
>> 1 1
```