Uploading and Reviewing the Data to Jupyter

In [1]:
```
import os
import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Set the path to the folder containing the CSV files
folder_path = '/Users/nisharams/Desktop/csv'

# Get a list of all files in the folder
file_list = os.listdir(folder_path)

# Filter only the CSV files from the list
csv_files = [file for file in file_list if file.endswith('.csv')]

# Initialize an empty dictionary to store tables and columns
tables_columns = {}

# Loop through each CSV file and extract tables and columns
for csv_file in csv_files:
    file_path = os.path.join(folder_path, csv_file)
    df = pd.read_csv(file_path)
    tables_columns[csv_file] = {'tables': [df], 'columns': list(df.columns

# Display the tables and columns information
for file_name, data in tables_columns.items():
    print(f"File: {file_name}")
    print(f"Tables: {data['tables']}")
    print(f"Columns: {data['columns']}")
    print("---")
```

```
File: hh_demographic.csv
Tables: [    age_desc marital_status_code income_desc homeowner_desc
hh_comp_desc  \
0        65+                  A      35-49K      HOMEOWNER  2 ADULTS N
O KIDS
1      45-54                  A      50-74K      HOMEOWNER  2 ADULTS N
O KIDS
2      25-34                  U      25-34K        UNKNOWN     2 ADULT
S KIDS
3      25-34                  U      75-99K      HOMEOWNER     2 ADULT
S KIDS
4      45-54                  B      50-74K      HOMEOWNER     SINGLE
FEMALE
..       ...                ...         ...            ...
...
796    35-44                  U      50-74K      HOMEOWNER  2 ADULTS N
O KIDS
797    45-54                  A      75-99K      HOMEOWNER           U
NKNOWN
```

Load the Data from CSV files

In [2]:
```
demographic_data = pd.read_csv('/Users/nisharams/Desktop/csv/hh_demograpl
weeks = pd.read_csv('/Users/nisharams/Desktop/csv/weeks.csv')
coupon = pd.read_csv('/Users/nisharams/Desktop/csv/coupon.csv')
campaign_table = pd.read_csv('/Users/nisharams/Desktop/csv/campaign_tabl
casual_data = pd.read_csv('/Users/nisharams/Desktop/csv/casual_data.csv'
coupon_redempt = pd.read_csv('/Users/nisharams/Desktop/csv/coupon_redemp
day_dates = pd.read_csv('/Users/nisharams/Desktop/csv/day_dates.csv')
product = pd.read_csv('/Users/nisharams/Desktop/csv/product.csv')
campaign_desc = pd.read_csv('/Users/nisharams/Desktop/csv/campaign_desc.
transaction_data = pd.read_csv('/Users/nisharams/Desktop/csv/transaction_
```

Check if all data has uploaded successfully

In [3]:
```
if all([demographic_data.shape[0] > 0, weeks.shape[0] > 0, coupon.shape[
        casual_data.shape[0] > 0, coupon_redempt.shape[0] > 0, day_dates
        campaign_desc.shape[0] > 0, transaction_data.shape[0] > 0]):
    print("All data has uploaded successfully")
```

```
All data has uploaded successfully
```

Coupon Optimization

Repeat Coupon Usage:

What proportion of customers are repeat coupon users?

-displays the count of customers who are repeat coupon users and those who are not which is helpful in understanding the customer loyalty and engagement with coupon promotions. If a significant proportion of customers are repeat coupon users, it indicates that the coupon strategy is effective in retaining customers.

In [4]:
```
# Calculate repeat coupon usage
repeat_coupon_users = coupon_redempt.groupby('household_key')['coupon_up
repeat_coupon_users = repeat_coupon_users.rename(columns={'coupon_upc':
repeat_coupon_users['is_repeat_user'] = repeat_coupon_users['num_coupons_
```

In [5]:
```
# Calculate repeat coupon usage
repeat_coupon_users = coupon_redempt.groupby('household_key')['coupon_up
repeat_coupon_users = repeat_coupon_users.rename(columns={'coupon_upc':
repeat_coupon_users['is_repeat_user'] = repeat_coupon_users['num_coupons_
```

In [6]:
```
# Merge demographic data with repeat coupon usage
demographic_data = pd.merge(demographic_data, repeat_coupon_users[['hous
```

In [7]:
```
# Visualization: Repeat Coupon Usage
plt.figure(figsize=(6, 6))
sns.countplot(data=demographic_data, x='is_repeat_user', color='red')
plt.xlabel('Repeat Coupon User')
plt.ylabel('Count')
plt.title('Repeat Coupon Usage')
sns.set_palette(["#b63a3a"])
plt.show()
```

The existing code calculates the proportion of customers who are repeat coupon users and those who are not. While this information is helpful in understanding customer loyalty and engagement, it does not directly address the objective of coupon optimization. To better fit the project's goal, we can modify the metric to calculate the repeat coupon usage as a percentage of total transactions, rather than just the count of customers. This will help in understanding how often customers are using coupons in their transactions and if there's a potential to reduce coupon usage while still acquiring customers.

In [8]:
```
# Calculate repeat coupon usage as a percentage of total transactions
total_transactions = len(transaction_data)
coupon_transactions = len(coupon_redempt)
repeat_coupon_usage_percentage = (coupon_transactions / total_transactio

# Display the result
print("Percentage of Repeat Coupon Usage: {:.2f}%".format(repeat_coupon_
```

```
Percentage of Repeat Coupon Usage: 0.15%
```

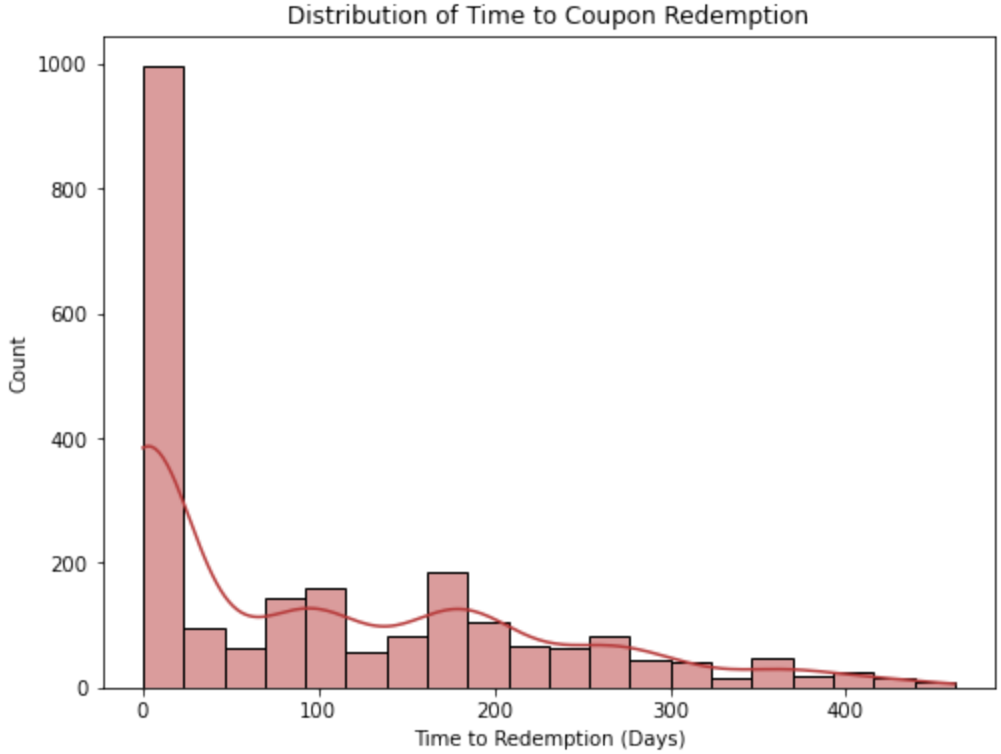
Calculate time to coupon redemption

What is the typical time it takes for customers to redeem coupons?

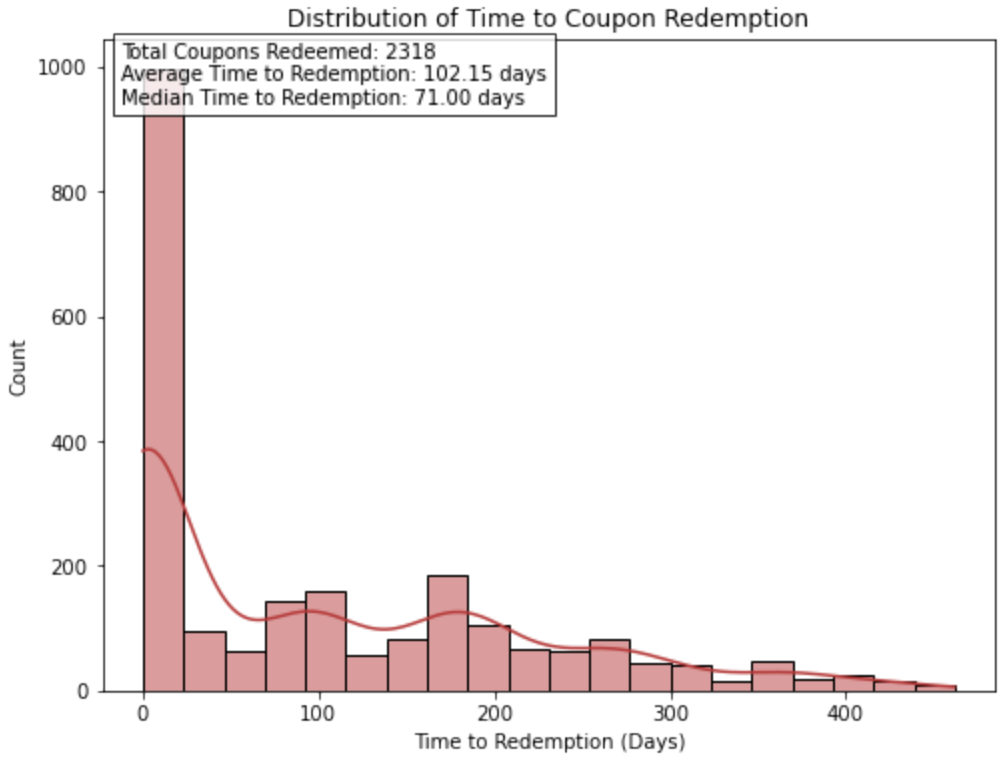
-The distribution of time it takes for customers to redeem coupons. The histogram shows the frequency of redemption times in days. From this plot, we can identify the typical time it takes for customers to redeem coupons. Shorter redemption times may indicate more successful and attractive coupon offers, while longer redemption times might imply less engaging or relevant coupon promotions.

In [9]:
```
# Calculate time to coupon redemption
coupon_redempt_time = coupon_redempt.groupby('household_key')['day'].min
coupon_redempt_time = coupon_redempt_time.rename(columns={'day': 'min_re
coupon_redempt = pd.merge(coupon_redempt, coupon_redempt_time, on='housel
coupon_redempt['time_to_redemption'] = coupon_redempt['day'] - coupon_re
```

In [10]:
```
# Visualization: Time to Coupon Redemption
plt.figure(figsize=(8, 6))  # Use the same figure size as Graph B
sns.histplot(data=coupon_redempt, x='time_to_redemption', bins=20, kde=T
plt.xlabel('Time to Redemption (Days)')
plt.ylabel('Count')
plt.title('Distribution of Time to Coupon Redemption')
sns.set_palette(["#b63a3a"])
plt.show()
```



In [12]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate time to coupon redemption
coupon_redempt_time = coupon_redempt.groupby('household_key')['day'].min
coupon_redempt_time = coupon_redempt_time.rename(columns={'day': 'min_re
coupon_redempt = pd.merge(coupon_redempt, coupon_redempt_time, on='housel
coupon_redempt['time_to_redemption'] = coupon_redempt['day'] - coupon_re

# Summary: Time to Coupon Redemption
total_coupons = len(coupon_redempt)  # Total number of coupons redeemed
average_time_to_redemption = coupon_redempt['time_to_redemption'].mean()
median_time_to_redemption = coupon_redempt['time_to_redemption'].median(

# Set the color palette before creating the plot
sns.set_palette(["#b63a3a"])

# Visualization: Time to Coupon Redemption
plt.figure(figsize=(8, 6))  # Use the same figure size as Graph B
sns.histplot(data=coupon_redempt, x='time_to_redemption', bins=20, kde=T
plt.xlabel('Time to Redemption (Days)')
plt.ylabel('Count')
plt.title('Distribution of Time to Coupon Redemption')

# Display the summary on the chart
summary_text = f"Total Coupons Redeemed: {total_coupons}\nAverage Time t
plt.text(0.02, 0.9, summary_text, transform=plt.gca().transAxes, bbox=di

plt.show()
```



Sales Uplift by Campaign Type:

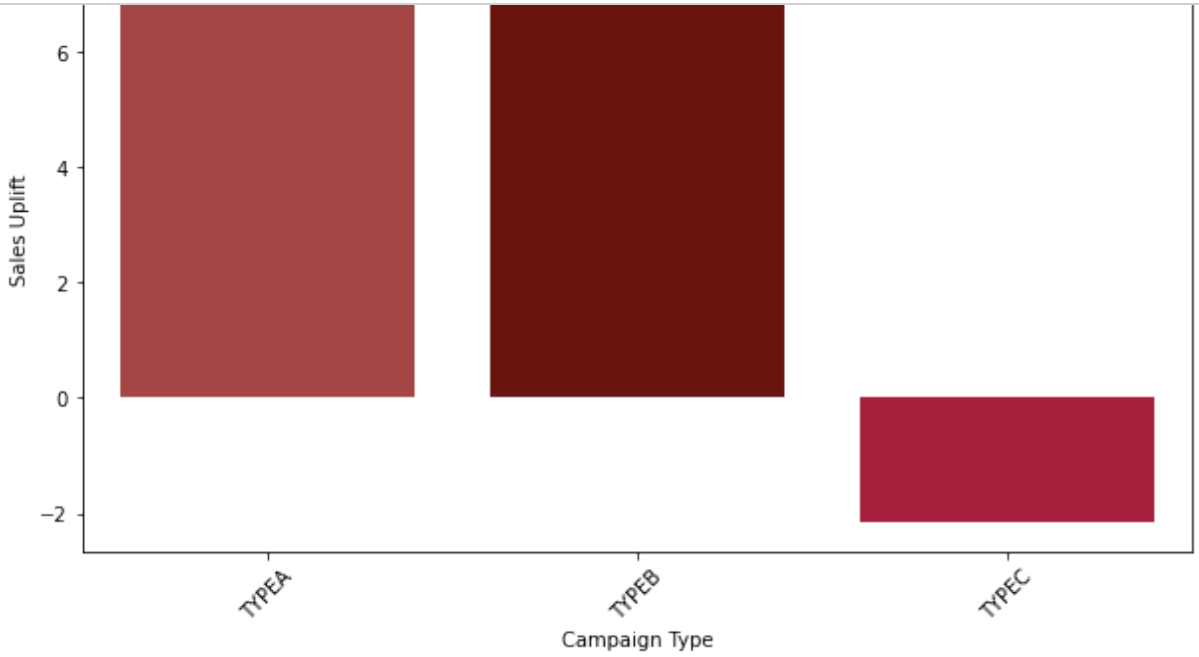
Helps identify which campaign types have the highest impact on increasing sales when customers use coupons. -we may consider eliminating campaigns of other types that are not performing as well in terms of sales uplift.

The bar plot shows the sales uplift for each campaign type attributed to the use of coupons. -x-axis = different campaign types (TYPEA, TYPEB, TYPEC, etc.), -y-axis = sales uplift in monetary values

In [13]:
```
# Merge campaign descriptions with transaction data
transaction_coupon_data = transaction_data.merge(campaign_table, on='hou
```

In [16]:
```
# Calculate Sales Uplift by Campaign Type
campaign_sales_uplift = transaction_coupon_data.groupby('description')['
campaign_sales_uplift = campaign_sales_uplift.reset_index()
```

In [17]:
```
# Visualization: Sales Uplift by Campaign Type
plt.figure(figsize=(10, 6))
sns.set_palette(["#b63a3a","#7c0a02","#BF0A30"])
sns.barplot(x='description', y='sales_value', data=campaign_sales_uplift
plt.title('Sales Uplift by Campaign Type')
plt.xlabel('Campaign Type')
plt.ylabel('Sales Uplift')
plt.xticks(rotation=45)
plt.show()
```



New Customer Acquisition Percentage

Determine the percentage of new customers who were acquired as a result of the coupon campaign. This metric assesses the campaign's impact on expanding the customer base.

In [18]:
```
# Calculate the total number of unique customers who made transactions du
total_customers = len(transaction_data['household_key'].unique())

# Calculate the number of unique customers who used coupons during the ca
coupon_customers = len(coupon_redempt['household_key'].unique())

# Calculate the percentage of new customers acquired as a result of the c
percentage_new_customers = (coupon_customers / total_customers) * 100

# Display the result
print("Percentage of New Customers Acquired: {:.2f}%".format(percentage_n
```

```
Percentage of New Customers Acquired: 18.99%
```

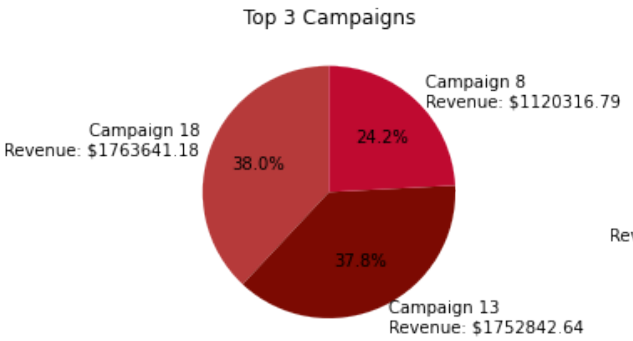
Evaluate the Overall Return on Investment:

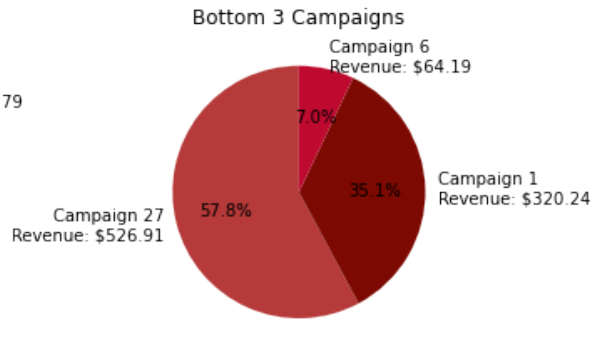
Provides a comprehensive view of the revenue generated from the coupon campaign compared to the cost of distributing the coupons. It helps the company assess the overall profitability of the campaign. If the return on investment is positive, it indicates that the coupon campaign is contributing to the company's bottom line. On the other hand, a negative return on investment would suggest that the coupon strategy needs adjustment or reconsideration.

In [19]:
```
import pandas as pd

# Load data from CSV files (assuming they are already loaded)
# campaign_table, coupon, product, and transaction_data

# Step 1: Get unique campaign IDs
campaign_ids = campaign_table['campaign'].unique()

# Step 2: Initialize a dictionary to store the revenue for each campaign
campaign_revenue = {}

# Step 3: Loop through each campaign to calculate revenue
for campaign_id in campaign_ids:
    # Step 4: Get the households participating in the campaign
    participating_households = campaign_table[campaign_table['campaign']

    # Step 5: Get the coupons associated with the campaign
    campaign_coupons = coupon[coupon['campaign'] == campaign_id]['coupon_

    # Step 6: Get the products eligible for redemption with the coupons
    eligible_products = coupon[coupon['coupon_upc'].isin(campaign_coupons

    # Step 7: Get the sales value of products redeemed in the campaign
    campaign_revenue[campaign_id] = transaction_data[transaction_data['h
                                                     transaction_data['p

# Sort the campaigns based on revenue from highest to lowest
sorted_campaign_revenue = sorted(campaign_revenue.items(), key=lambda x:

# Display the sorted revenue for each campaign
for campaign_id, revenue in sorted_campaign_revenue:
    print(f"Campaign {campaign_id}: Revenue = ${revenue:.2f}")
```

```
Campaign 18: Revenue = $1763641.18
Campaign 13: Revenue = $1752842.64
Campaign 8: Revenue = $1120316.79
Campaign 30: Revenue = $65025.53
Campaign 26: Revenue = $64629.15
Campaign 22: Revenue = $25163.82
Campaign 25: Revenue = $23376.12
Campaign 23: Revenue = $18977.59
Campaign 24: Revenue = $17904.39
Campaign 17: Revenue = $17579.72
Campaign 9: Revenue = $15516.97
Campaign 19: Revenue = $14139.87
Campaign 12: Revenue = $13361.48
Campaign 10: Revenue = $12959.59
Campaign 5: Revenue = $12200.55
Campaign 14: Revenue = $9726.20
Campaign 20: Revenue = $8972.84
Campaign 29: Revenue = $7560.99
Campaign 16: Revenue = $5823.58
Campaign 11: Revenue = $3547.60
Campaign 7: Revenue = $2777.03
Campaign 21: Revenue = $2623.36
Campaign 4: Revenue = $1888.88
Campaign 2: Revenue = $1550.55
Campaign 28: Revenue = $1448.63
Campaign 15: Revenue = $747.57
Campaign 3: Revenue = $667.73
Campaign 27: Revenue = $526.91
Campaign 1: Revenue = $320.24
Campaign 6: Revenue = $64.19
```

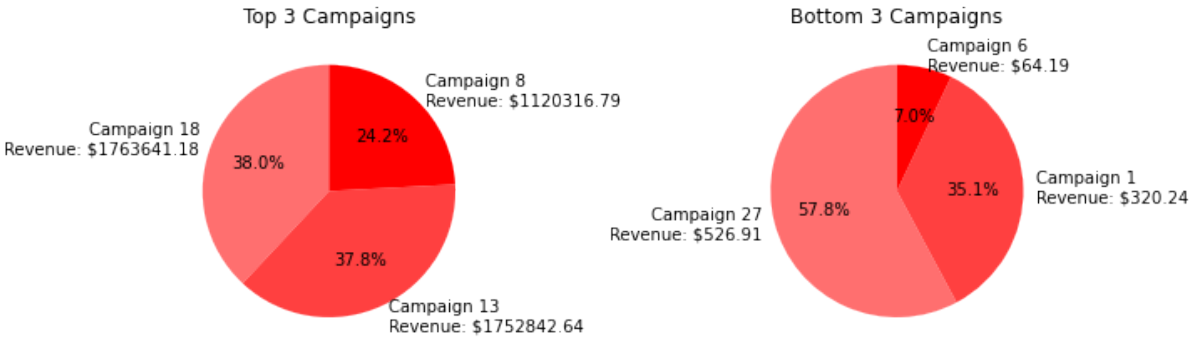
In [20]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data from CSV files (assuming they are already loaded)
# campaign_table, coupon, product, and transaction_data

# Step 1: Get unique campaign IDs
campaign_ids = campaign_table['campaign'].unique()

# Step 2: Initialize a dictionary to store the revenue for each campaign
campaign_revenue = {}

# Step 3: Loop through each campaign to calculate revenue
for campaign_id in campaign_ids:
    # Step 4: Get the households participating in the campaign
    participating_households = campaign_table[campaign_table['campaign']

    # Step 5: Get the coupons associated with the campaign
    campaign_coupons = coupon[coupon['campaign'] == campaign_id]['coupon_

    # Step 6: Get the products eligible for redemption with the coupons
    eligible_products = coupon[coupon['coupon_upc'].isin(campaign_coupons

    # Step 7: Get the sales value of products redeemed in the campaign
    campaign_revenue[campaign_id] = transaction_data[transaction_data['h
                                                     transaction_data['p

# Sort the campaigns based on revenue from highest to lowest
sorted_campaign_revenue = sorted(campaign_revenue.items(), key=lambda x:

# Get the top 3 and bottom 3 campaigns based on revenue
top_3_campaigns = sorted_campaign_revenue[:3]
bottom_3_campaigns = sorted_campaign_revenue[-3:]

# Create labels and sizes for the pie charts
top_labels = [f"Campaign {campaign_id}\nRevenue: ${revenue:.2f}" for camp
top_sizes = [revenue for _, revenue in top_3_campaigns]

bottom_labels = [f"Campaign {campaign_id}\nRevenue: ${revenue:.2f}" for
bottom_sizes = [revenue for _, revenue in bottom_3_campaigns]

# Create two subplots for the pie charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Plot the top 3 campaigns pie chart
ax1.pie(top_sizes, labels=top_labels, autopct='%1.1f%%', startangle=90)
ax1.set_title('Top 3 Campaigns')

# Plot the bottom 3 campaigns pie chart
ax2.pie(bottom_sizes, labels=bottom_labels, autopct='%1.1f%%', startangl
ax2.set_title('Bottom 3 Campaigns')

plt.tight_layout()
plt.show()
```


Top 3 Campaigns
Campaign 8
Revenue: $1120316.79
Campaign 18
Revenue: $1763641.18
Campaign 13
Revenue: $1752842.64
24.2%
38.0%
37.8%

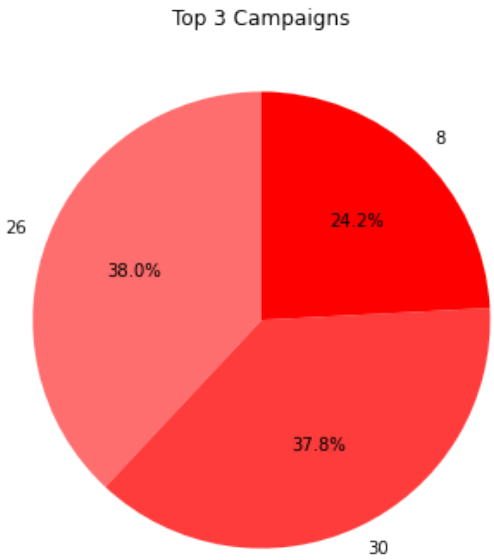

Bottom 3 Campaigns
Campaign 6
Revenue: $64.19
Campaign 1
Revenue: $320.24
Campaign 27
Revenue: $526.91
7.0%
35.1%
57.8%

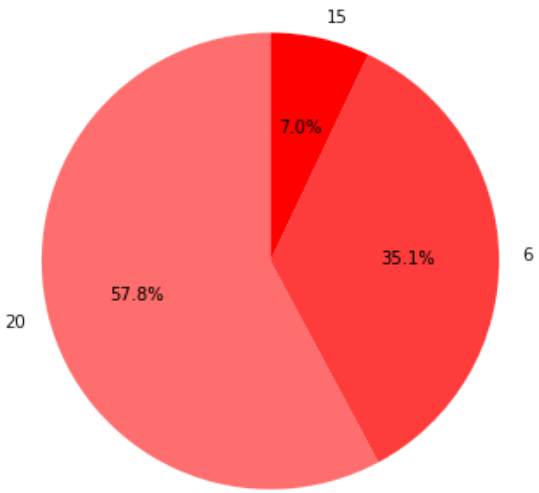
In [21]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data from CSV files (assuming they are already loaded)
# campaign_table, coupon, product, and transaction_data

# Step 1: Get unique campaign IDs
campaign_ids = campaign_table['campaign'].unique()

# Step 2: Initialize a dictionary to store the revenue for each campaign
campaign_revenue = {}

# Step 3: Loop through each campaign to calculate revenue
for campaign_id in campaign_ids:
    # Step 4: Get the households participating in the campaign
    participating_households = campaign_table[campaign_table['campaign']

    # Step 5: Get the coupons associated with the campaign
    campaign_coupons = coupon[coupon['campaign'] == campaign_id]['coupon_

    # Step 6: Get the products eligible for redemption with the coupons
    eligible_products = coupon[coupon['coupon_upc'].isin(campaign_coupons

    # Step 7: Get the sales value of products redeemed in the campaign
    campaign_revenue[campaign_id] = transaction_data[transaction_data['h
                                                     transaction_data['p

# Sort the campaigns based on revenue from highest to lowest
sorted_campaign_revenue = sorted(campaign_revenue.items(), key=lambda x:

# Get the top 3 and bottom 3 campaigns based on revenue
top_3_campaigns = sorted_campaign_revenue[:3]
bottom_3_campaigns = sorted_campaign_revenue[-3:]

# Create labels and sizes for the pie charts
top_labels = [f"Campaign {campaign_id}\nRevenue: ${revenue:.2f}" for camp
top_sizes = [revenue for _, revenue in top_3_campaigns]

bottom_labels = [f"Campaign {campaign_id}\nRevenue: ${revenue:.2f}" for
bottom_sizes = [revenue for _, revenue in bottom_3_campaigns]

# Define colors in the red family
red_colors = ['#FF6F6F', '#FF4040', '#FF0000']

# Create two subplots for the pie charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Plot the top 3 campaigns pie chart
ax1.pie(top_sizes, labels=top_labels, autopct='%1.1f%%', startangle=90, 
ax1.set_title('Top 3 Campaigns')

# Plot the bottom 3 campaigns pie chart
ax2.pie(bottom_sizes, labels=bottom_labels, autopct='%1.1f%%', startangl
ax2.set_title('Bottom 3 Campaigns')

plt.tight_layout()
```

```
plt.show()
```

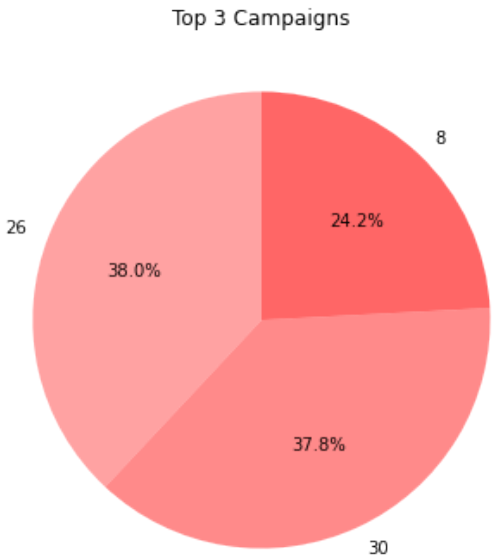

Top 3 Campaigns
Campaign 8
Revenue: $1120316.79
Campaign 18
Revenue: $1763641.18
Campaign 13
Revenue: $1752842.64
24.2%
38.0%
37.8%
Bottom 3 Campaigns
Campaign 6
Revenue: $64.19
Campaign 1
Revenue: $320.24
Campaign 27
Revenue: $526.91
7.0%
35.1%
57.8%

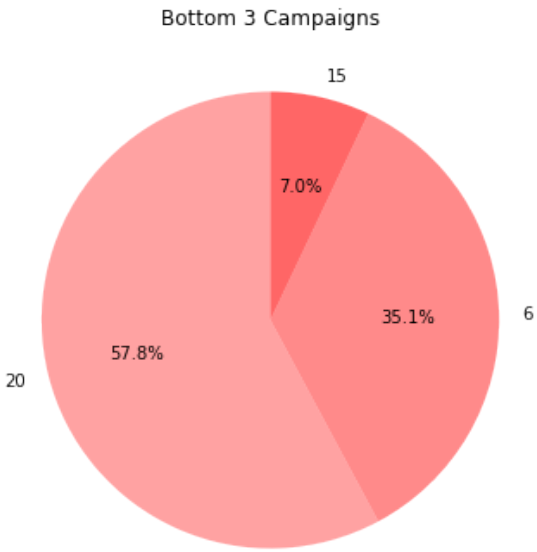
In [22]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data from CSV files (assuming they are already loaded)
# campaign_table, coupon, product, and transaction_data

# Step 1: Get unique campaign IDs
campaign_ids = campaign_table['campaign'].unique()

# Step 2: Initialize a dictionary to store the revenue for each campaign
campaign_revenue = {}

# Step 3: Loop through each campaign to calculate revenue
for campaign_id in campaign_ids:
    # Step 4: Get the households participating in the campaign
    participating_households = campaign_table[campaign_table['campaign']

    # Step 5: Get the coupons associated with the campaign
    campaign_coupons = coupon[coupon['campaign'] == campaign_id]['coupon_

    # Step 6: Get the products eligible for redemption with the coupons
    eligible_products = coupon[coupon['coupon_upc'].isin(campaign_coupons

    # Step 7: Get the sales value of products redeemed in the campaign
    campaign_revenue[campaign_id] = transaction_data[transaction_data['h
                                                     transaction_data['p

# Sort the campaigns based on revenue from highest to lowest
sorted_campaign_revenue = sorted(campaign_revenue.items(), key=lambda x:

# Get the top 3 and bottom 3 campaigns based on revenue
top_3_campaigns = sorted_campaign_revenue[:3]
bottom_3_campaigns = sorted_campaign_revenue[-3:]

# Create sizes for the pie charts
top_sizes = [revenue for _, revenue in top_3_campaigns]
bottom_sizes = [revenue for _, revenue in bottom_3_campaigns]

# Define colors in the red family
red_colors = ['#FF6F6F', '#FF4040', '#FF0000']

# Create two subplots for the pie charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Plot the top 3 campaigns pie chart with campaign numbers inside
ax1.pie(top_sizes, labels=campaign_ids[:3], autopct='%1.1f%%', startangl
ax1.set_title('Top 3 Campaigns')

# Plot the bottom 3 campaigns pie chart with campaign numbers inside
ax2.pie(bottom_sizes, labels=campaign_ids[-3:], autopct='%1.1f%%', start
ax2.set_title('Bottom 3 Campaigns')

plt.tight_layout()
plt.show()
```


Top 3 Campaigns
8
24.2%
26
38.0%
37.8%
30


Bottom 3 Campaigns
15
7.0%
35.1%
6
57.8%
20

In [24]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data from CSV files (assuming they are already loaded)
# campaign_table, coupon, product, and transaction_data

# Step 1: Get unique campaign IDs
campaign_ids = campaign_table['campaign'].unique()

# Step 2: Initialize a dictionary to store the revenue for each campaign
campaign_revenue = {}

# Step 3: Loop through each campaign to calculate revenue
for campaign_id in campaign_ids:
    # Step 4: Get the households participating in the campaign
    participating_households = campaign_table[campaign_table['campaign']

    # Step 5: Get the coupons associated with the campaign
    campaign_coupons = coupon[coupon['campaign'] == campaign_id]['coupon_

    # Step 6: Get the products eligible for redemption with the coupons
    eligible_products = coupon[coupon['coupon_upc'].isin(campaign_coupons

    # Step 7: Get the sales value of products redeemed in the campaign
    campaign_revenue[campaign_id] = transaction_data[transaction_data['h
                                                     transaction_data['p

# Sort the campaigns based on revenue from highest to lowest
sorted_campaign_revenue = sorted(campaign_revenue.items(), key=lambda x:

# Get the top 3 and bottom 3 campaigns based on revenue
top_3_campaigns = sorted_campaign_revenue[:3]
bottom_3_campaigns = sorted_campaign_revenue[-3:]

# Create sizes for the pie charts
top_sizes = [revenue for _, revenue in top_3_campaigns]
bottom_sizes = [revenue for _, revenue in bottom_3_campaigns]

# Define colors in the red family with reduced alpha value
red_colors = [(1.0, 0.4, 0.4, 0.6), (1.0, 0.25, 0.25, 0.6), (1.0, 0.0, 0

# Create two subplots for the pie charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Plot the top 3 campaigns pie chart with campaign numbers inside
ax1.pie(top_sizes, labels=campaign_ids[:3], autopct='%1.1f%%', startangl
ax1.set_title('Top 3 Campaigns')

# Plot the bottom 3 campaigns pie chart with campaign numbers inside
ax2.pie(bottom_sizes, labels=campaign_ids[-3:], autopct='%1.1f%%', start
ax2.set_title('Bottom 3 Campaigns')

plt.tight_layout()
plt.show()
```





Demographics

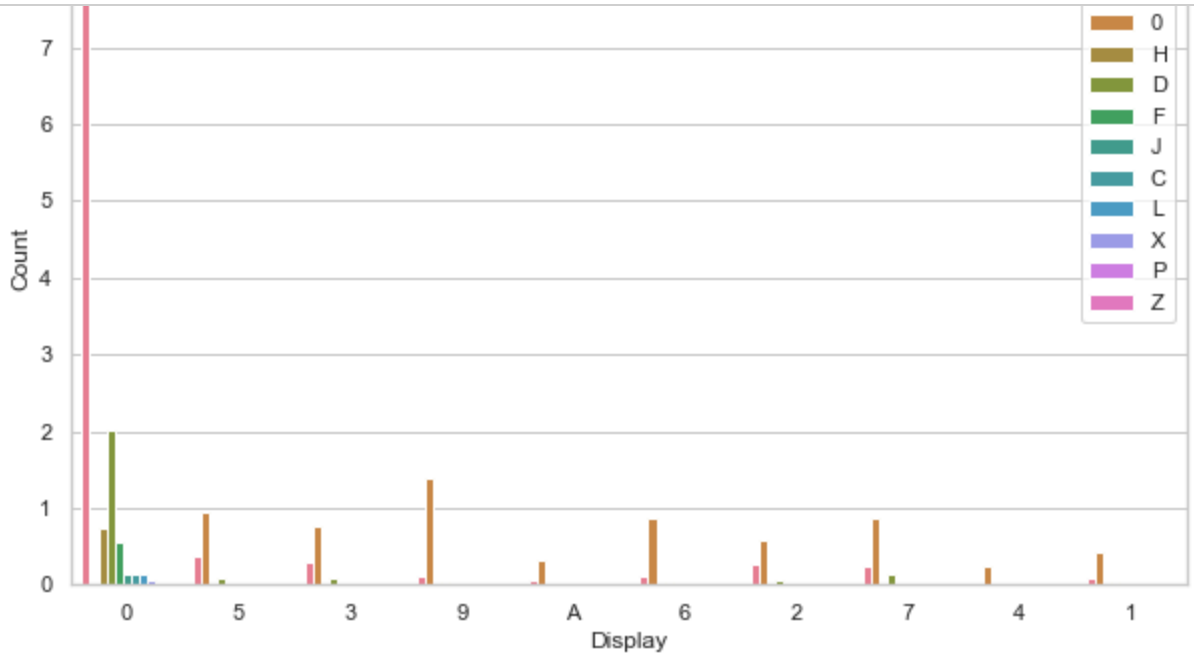
In [25]:
```
import pandas as pd

# Load data from CSV files
demographic_data = pd.read_csv('/Users/nisharams/Desktop/csv/hh_demograpl
weeks = pd.read_csv('/Users/nisharams/Desktop/csv/weeks.csv')
coupon = pd.read_csv('/Users/nisharams/Desktop/csv/coupon.csv')
campaign_table = pd.read_csv('/Users/nisharams/Desktop/csv/campaign_tabl
casual_data = pd.read_csv('/Users/nisharams/Desktop/csv/casual_data.csv'
coupon_redempt = pd.read_csv('/Users/nisharams/Desktop/csv/coupon_redemp
day_dates = pd.read_csv('/Users/nisharams/Desktop/csv/day_dates.csv')
product = pd.read_csv('/Users/nisharams/Desktop/csv/product.csv')
campaign_desc = pd.read_csv('/Users/nisharams/Desktop/csv/campaign_desc.
transaction_data = pd.read_csv('/Users/nisharams/Desktop/csv/transaction_

# Summary of key insights

# 1. Demographics of Customers
print("Demographic Data:")
print(demographic_data.head())

# 2. Campaigns and Coupons
print("\nCampaign Table:")
print(campaign_table.head())
print("\nCoupon Table:")
print(coupon.head())

# 3. Product Data
print("\nProduct Data:")
print(product.head())

# 4. Transaction Data
print("\nTransaction Data:")
print(transaction_data.head())

# 5. Coupon Redemption
print("\nCoupon Redemption Data:")
print(coupon_redempt.head())

# 6. Causal Data
print("\nCausal Data:")
print(casual_data.head())
```

```
Demographic Data:
  age_desc marital_status_code income_desc homeowner_desc      hh_comp_
desc  \
0      65+                   A      35-49K      HOMEOWNER  2 ADULTS NO
KIDS
1    45-54                   A      50-74K      HOMEOWNER  2 ADULTS NO
KIDS
2    25-34                   U      25-34K        UNKNOWN     2 ADULTS
KIDS
3    25-34                   U      75-99K      HOMEOWNER     2 ADULTS
KIDS
4    45-54                   B      50-74K      HOMEOWNER     SINGLE FE
MALE

  household_size_desc kid_category_desc  household_key  in_data
0                   2      NONE/UNKNOWN              1        1
1                   2      NONE/UNKNOWN              7        1
2                   3                 1              8        1
3                   4                 2             13        1
4                   1      NONE/UNKNOWN             16        0

Campaign Table:
  description  household_key  campaign
0       TYPEA             17        26
1       TYPEA             27        26
2       TYPEA            212        26
3       TYPEA            208        26
4       TYPEA            192        26

Coupon Table:
    coupon_upc  product_id  campaign
0  10000089061       27160         4
1  10000089064       27754         9
2  10000089073       28897        12
3  51800009050       28919        28
4  52100000076       28929        25

Product Data:
   product_id  manufacturer    department     brand            commodit
y_desc  \
0       25671             2       GROCERY  NATIONAL                  FR
ZN ICE
1       26081             2  MISC. TRANS.  NATIONAL  NO COMMODITY DESCR
IPTION
2       26093            69        PASTRY   PRIVATE
BREAD
3       26190            69       GROCERY   PRIVATE      FRUIT - SHELF
STABLE
4       26355            69       GROCERY   PRIVATE             COOKIE
S/CONES

            sub_commodity_desc curr_size_of_product  in_data
0          ICE - CRUSHED/CUBED                22 LB        0
1  NO SUBCOMMODITY DESCRIPTION                  NaN        0
2         BREAD:ITALIAN/FRENCH                  NaN        0
3                  APPLE SAUCE                50 OZ        0
4            SPECIALTY COOKIES                14 OZ        0
```

```
Transaction Data:
   household_key    basket_id  day  product_id  quantity  sales_value
\
0           2375  26984851472    1     1004906         1         1.39
1           2375  26984851472    1     1033142         1         0.82
2           2375  26984851472    1     1036325         1         0.99
3           2375  26984851472    1     1082185         1         1.21
4           2375  26984851472    1     8160430         1         1.50

   store_id  retail_disc  trans_time  week_no  coupon_disc  coupon_matc
h_disc
0       364        -0.60        1631        1          0.0
0.0
1       364         0.00        1631        1          0.0
0.0
2       364        -0.30        1631        1          0.0
0.0
3       364         0.00        1631        1          0.0
0.0
4       364        -0.39        1631        1          0.0
0.0

Coupon Redemption Data:
   household_key  day   coupon_upc  campaign
0              1  421  10000085364         8
1              1  421  51700010076         8
2              1  427  54200000033         8
3              1  597  10000085476        18
4              1  597  54200029176        18

Causal Data:
   product_id  store_id  week_no display mailer
0       26190       288       70       0      A
1       26190       292       70       0      A
2       26190       295       70       0      A
3       26190       296       70       0      A
4       26190       298       70       0      A
```

In [27]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load data from CSV files
# (Assuming the file paths have been updated correctly as mentioned in t

# 1. Demographics of Customers
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.countplot(data=demographic_data, x='age_desc', hue='marital_status_c
plt.title('Distribution of Age by Marital Status')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Marital Status', loc='upper right')
plt.show()

# 2. Campaigns and Coupons
plt.figure(figsize=(10, 6))
sns.countplot(data=campaign_table, x='description', palette='viridis')
plt.title('Distribution of Campaign Types')
plt.xlabel('Campaign Type')
plt.ylabel('Count')
plt.show()

# 3. Product Data
plt.figure(figsize=(12, 6))
sns.countplot(data=product, x='department', palette='Set2')
plt.title('Distribution of Products by Department')
plt.xlabel('Department')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()

# 4. Transaction Data
plt.figure(figsize=(10, 6))
sns.histplot(data=transaction_data, x='sales_value', bins=50, kde=True, 
plt.title('Distribution of Sales Value')
plt.xlabel('Sales Value')
plt.ylabel('Frequency')
plt.show()

# 5. Coupon Redemption
plt.figure(figsize=(10, 6))
sns.countplot(data=coupon_redempt, x='campaign', palette='Paired')
plt.title('Coupon Redemption by Campaign')
plt.xlabel('Campaign')
plt.ylabel('Count')
plt.show()

# 6. Causal Data
plt.figure(figsize=(10, 6))
sns.countplot(data=casual_data, x='display', hue='mailer', palette='husl
plt.title('Causal Data - Product Display and Mailer')
plt.xlabel('Display')
plt.ylabel('Count')
plt.legend(title='Mailer', loc='upper right')
```

```
plt.show()
```



In [ ]:

```
```

In [ ]:

```
lists of the top and bottom 3 customer demographic segments based on so
```

In [28]:
```
# Assuming demographic_data is already loaded

# Get a list of demographic columns
demographic_columns = ['age_desc', 'marital_status_code', 'income_desc',
                       'hh_comp_desc', 'household_size_desc', 'kid_categ

# Create dictionaries to store top and bottom segments for each demograp
top_segments_dict = {}
bottom_segments_dict = {}

# Loop through each demographic column and find the top and bottom segme
for column in demographic_columns:
    top_segments = demographic_data[column].value_counts().nlargest(3).i
    bottom_segments = demographic_data[column].value_counts().nsmallest(
    top_segments_dict[column] = top_segments
    bottom_segments_dict[column] = bottom_segments

# Print the results
for column in demographic_columns:
    print(f"\nTop 3 customer demographic segments for '{column}':")
    print(top_segments_dict[column])

    print(f"\nBottom 3 customer demographic segments for '{column}':")
    print(bottom_segments_dict[column])
```

```
Top 3 customer demographic segments for 'age_desc':
['45-54', '35-44', '25-34']

Bottom 3 customer demographic segments for 'age_desc':
['19-24', '55-64', '65+']

Top 3 customer demographic segments for 'marital_status_code':
['U', 'A', 'B']

Bottom 3 customer demographic segments for 'marital_status_code':
['B', 'A', 'U']

Top 3 customer demographic segments for 'income_desc':
['50-74K', '35-49K', '75-99K']

Bottom 3 customer demographic segments for 'income_desc':
['200-249K', '250K+', '175-199K']

Top 3 customer demographic segments for 'homeowner_desc':
['HOMEOWNER', 'UNKNOWN', 'RENTER']

Bottom 3 customer demographic segments for 'homeowner_desc':
['PROBABLE RENTER', 'PROBABLE OWNER', 'RENTER']

Top 3 customer demographic segments for 'hh_comp_desc':
['2 ADULTS NO KIDS', '2 ADULTS KIDS', 'SINGLE FEMALE']

Bottom 3 customer demographic segments for 'hh_comp_desc':
['1 ADULT KIDS', 'UNKNOWN', 'SINGLE MALE']

Top 3 customer demographic segments for 'household_size_desc':
['2', '1', '3']

Bottom 3 customer demographic segments for 'household_size_desc':
['4', '5+', '3']

Top 3 customer demographic segments for 'kid_category_desc':
['NONE/UNKNOWN', '1', '3+']

Bottom 3 customer demographic segments for 'kid_category_desc':
['2', '3+', '1']
```

In [29]:
```
# Assuming demographic_data is already loaded

import pandas as pd

# Get a list of demographic columns
demographic_columns = ['age_desc', 'marital_status_code', 'income_desc',
                       'hh_comp_desc', 'household_size_desc', 'kid_categ

# Create dictionaries to store top and bottom segments for each demograp
top_segments_dict = {}
bottom_segments_dict = {}

# Loop through each demographic column and find the top and bottom segme
for column in demographic_columns:
    top_segments = demographic_data[column].value_counts().nlargest(3).i
    bottom_segments = demographic_data[column].value_counts().nsmallest(
    top_segments_dict[column] = top_segments
    bottom_segments_dict[column] = bottom_segments

# Create DataFrames for top and bottom segments
top_df = pd.DataFrame(top_segments_dict)
bottom_df = pd.DataFrame(bottom_segments_dict)

# Concatenate DataFrames side by side
result_table = pd.concat([top_df.add_prefix("Top_"), bottom_df.add_prefi

# Print the result table
print(result_table)
```

```
  Top_age_desc Top_marital_status_code Top_income_desc Top_homeowner_de
sc  \
0        45-54                       U          50-74K          HOMEOWN
ER
1        35-44                       A          35-49K            UNKNO
WN
2        25-34                       B          75-99K             RENT
ER

   Top_hh_comp_desc Top_household_size_desc Top_kid_category_desc  \
0  2 ADULTS NO KIDS                       2          NONE/UNKNOWN
1     2 ADULTS KIDS                       1                     1
2     SINGLE FEMALE                       3                    3+

  Bottom_age_desc Bottom_marital_status_code Bottom_income_desc  \
0           19-24                          B           200-249K
1           55-64                          A              250K+
2             65+                          U           175-199K

  Bottom_homeowner_desc Bottom_hh_comp_desc Bottom_household_size_desc
\
0       PROBABLE RENTER        1 ADULT KIDS                          4
1        PROBABLE OWNER             UNKNOWN                         5+
2                RENTER         SINGLE MALE                          3

  Bottom_kid_category_desc
0                        2
1                       3+
2                        1
```

In [30]:
```
import pandas as pd

# Load data from CSV files
demographic_data = pd.read_csv('/Users/nisharams/Desktop/csv/hh_demograpl

# Calculate visit percentages for each demographic segment
visit_percentage = demographic_data.groupby(['age_desc', 'marital_status_

# Calculate total count for each demographic segment
visit_percentage['total_count'] = visit_percentage.iloc[:, -2:].sum(axis=

# Calculate visit percentage for each demographic segment
visit_percentage['visit_percentage'] = (visit_percentage[1] / visit_perc

# Sort by visit_percentage
visit_percentage = visit_percentage.sort_values(by='visit_percentage', as

# Get the top 3 and bottom 3 demographic segments based on visit_percenta
top_3_segments = visit_percentage.head(3)
bottom_3_segments = visit_percentage.tail(3)

# Display the results
print("Top 3 Customer Demographic Segments (Visiting the Store):")
print(top_3_segments[['age_desc', 'marital_status_code', 'income_desc',

print("\nBottom 3 Customer Demographic Segments (Not Visiting the Store)
print(bottom_3_segments[['age_desc', 'marital_status_code', 'income_desc
```

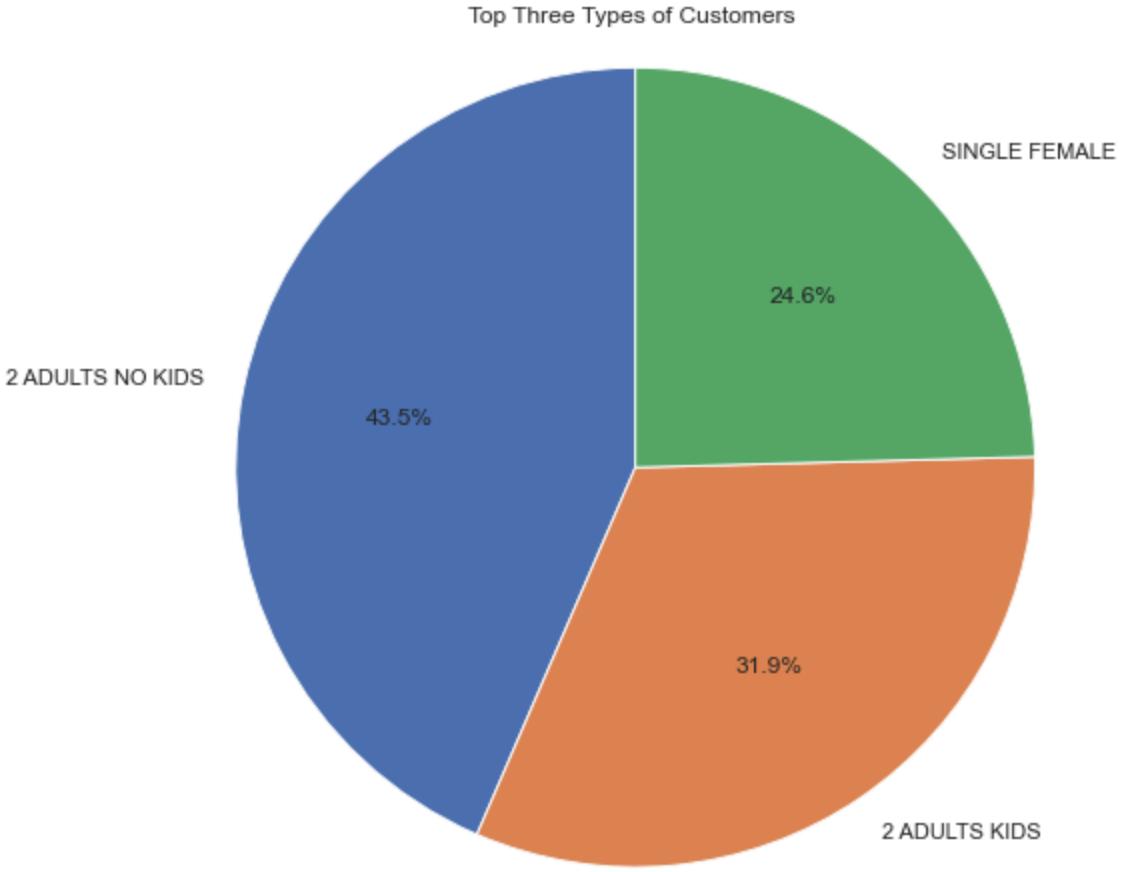
```
Top 3 Customer Demographic Segments (Visiting the Store):
in_data age_desc marital_status_code income_desc homeowner_desc  \
0          19-24                   A      15-24K      HOMEOWNER
241        35-44                   U      35-49K        UNKNOWN
337        45-54                   U    100-124K        UNKNOWN

in_data   hh_comp_desc household_size_desc kid_category_desc  visit_per
centage
0        2 ADULTS KIDS                   3                 1
100.0
241      2 ADULTS KIDS                   3                 1
100.0
337      SINGLE FEMALE                   1      NONE/UNKNOWN
100.0

Bottom 3 Customer Demographic Segments (Not Visiting the Store):
in_data age_desc marital_status_code income_desc homeowner_desc  \
213        35-44                   B      50-74K      HOMEOWNER
474          65+                   B      15-24K      HOMEOWNER
13         19-24                   B      50-74K        UNKNOWN

in_data      hh_comp_desc household_size_desc kid_category_desc  \
213      2 ADULTS NO KIDS                   2      NONE/UNKNOWN
474         SINGLE FEMALE                   1      NONE/UNKNOWN
13           2 ADULTS KIDS                  3                 1

in_data  visit_percentage
213                   0.0
474                   0.0
13                    0.0
```

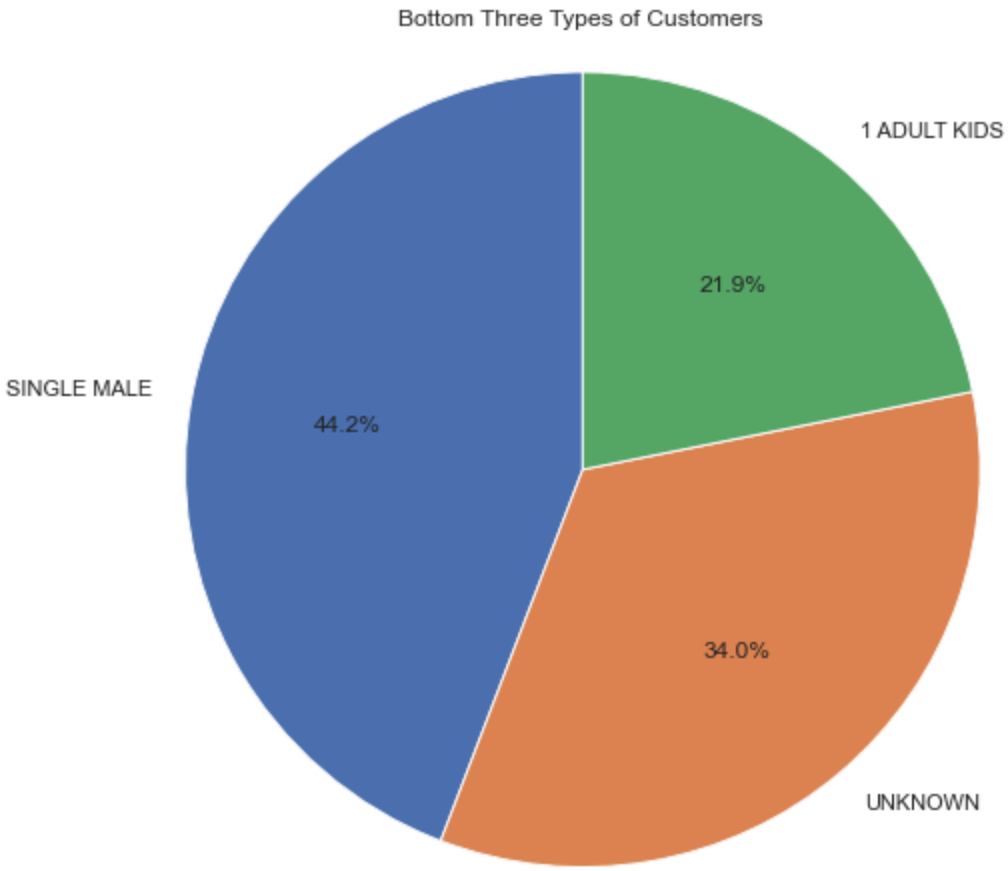
In [31]:
```
# Import necessary libraries
import pandas as pd

# Load the demographic data from the CSV file
demographic_data = pd.read_csv('/Users/nisharams/Desktop/csv/hh_demograpl

# Summarize the key insights about customer demographics
# 1. Age distribution
age_distribution = demographic_data['age_desc'].value_counts()

# 2. Marital status distribution
marital_status_distribution = demographic_data['marital_status_code'].va

# 3. Income distribution
income_distribution = demographic_data['income_desc'].value_counts()

# 4. Homeownership distribution
homeownership_distribution = demographic_data['homeowner_desc'].value_co

# 5. Household composition distribution
household_composition_distribution = demographic_data['hh_comp_desc'].va

# 6. Household size distribution
household_size_distribution = demographic_data['household_size_desc'].va

# 7. Kid category distribution
kid_category_distribution = demographic_data['kid_category_desc'].value_

# Display the key insights
print("Age Distribution:")
print(age_distribution)

print("\nMarital Status Distribution:")
print(marital_status_distribution)

print("\nIncome Distribution:")
print(income_distribution)

print("\nHomeownership Distribution:")
print(homeownership_distribution)

print("\nHousehold Composition Distribution:")
print(household_composition_distribution)

print("\nHousehold Size Distribution:")
print(household_size_distribution)

print("\nKid Category Distribution:")
print(kid_category_distribution)
```

```
Age Distribution:
45-54    288
35-44    194
25-34    142
65+       72
55-64     59
19-24     46
Name: age_desc, dtype: int64

Marital Status Distribution:
U    344
A    340
B    117
Name: marital_status_code, dtype: int64

Income Distribution:
50-74K       192
35-49K       172
75-99K        96
25-34K        77
15-24K        74
UNDER 15K     61
125-149K      38
100-124K      34
150-174K      30
250K+         11
175-199K      11
200-249K       5
Name: income_desc, dtype: int64

Homeownership Distribution:
HOMEOWNER          504
UNKNOWN            233
RENTER              42
PROBABLE RENTER     11
PROBABLE OWNER      11
Name: homeowner_desc, dtype: int64

Household Composition Distribution:
2 ADULTS NO KIDS    255
2 ADULTS KIDS       187
SINGLE FEMALE       144
SINGLE MALE          95
UNKNOWN              73
1 ADULT KIDS         47
Name: hh_comp_desc, dtype: int64

Household Size Distribution:
2     318
1     255
3     109
5+     66
4      53
Name: household_size_desc, dtype: int64

Kid Category Distribution:
NONE/UNKNOWN    558
```

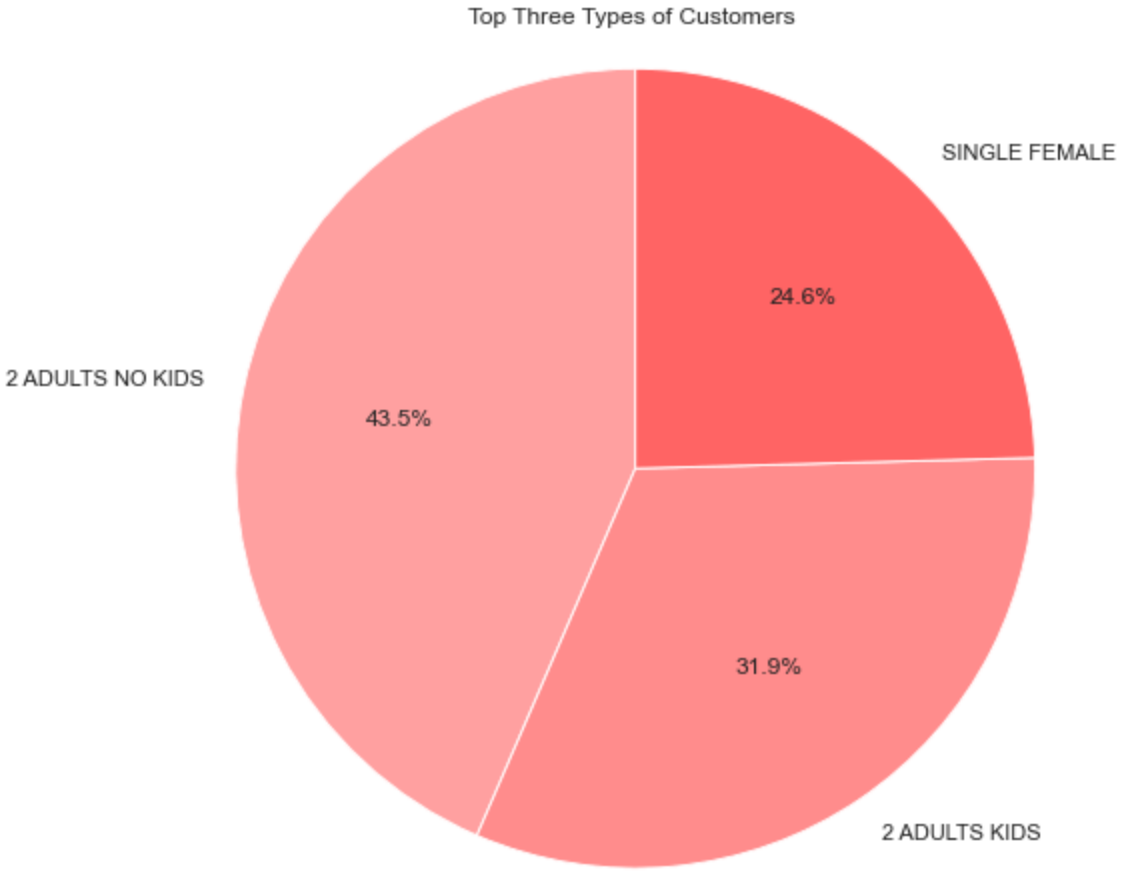
```
1               114
3+               69
2                60
Name: kid_category_desc, dtype: int64
```

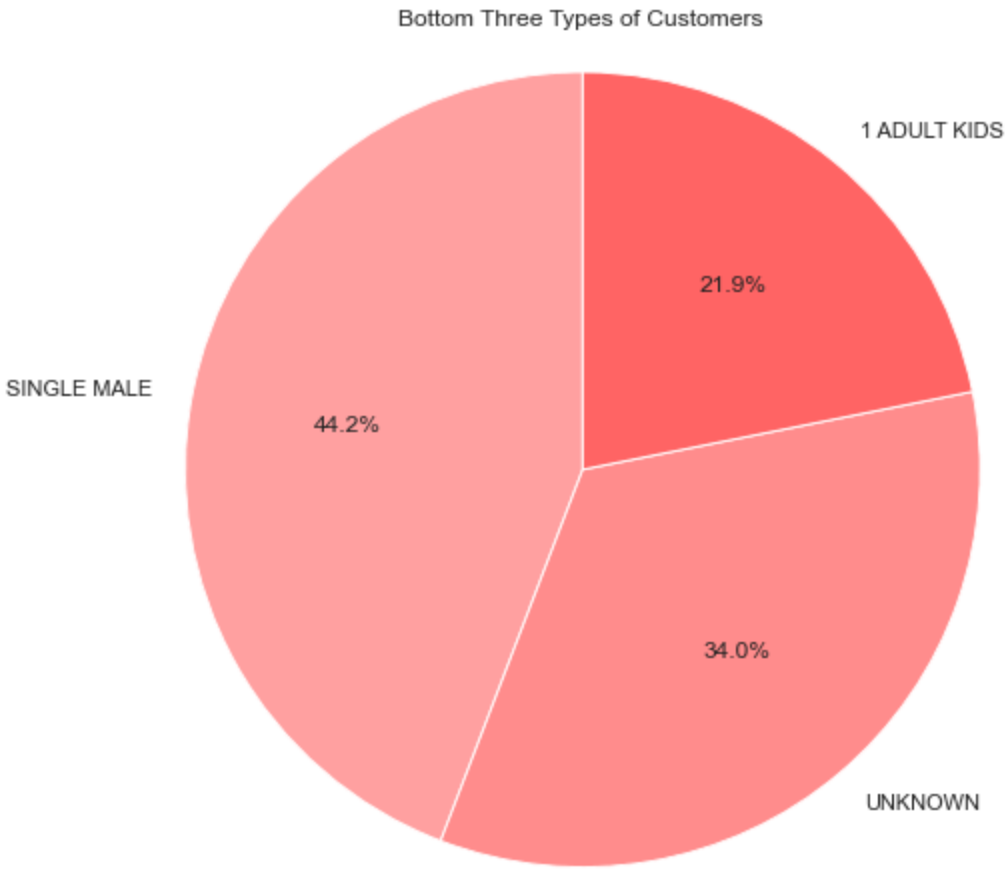
In [32]:
```
import matplotlib.pyplot as plt

# Calculate the percentage of each type of customer
customer_counts = demographic_data['hh_comp_desc'].value_counts(normaliz
top_three_customers = customer_counts.head(3)
bottom_three_customers = customer_counts.tail(3)

# Pie chart for top three customers
plt.figure(figsize=(8, 8))
plt.pie(top_three_customers, labels=top_three_customers.index, autopct='
plt.title('Top Three Types of Customers')
plt.axis('equal')
plt.show()

# Pie chart for bottom three customers
plt.figure(figsize=(8, 8))
plt.pie(bottom_three_customers, labels=bottom_three_customers.index, aut
plt.title('Bottom Three Types of Customers')
plt.axis('equal')
plt.show()
```



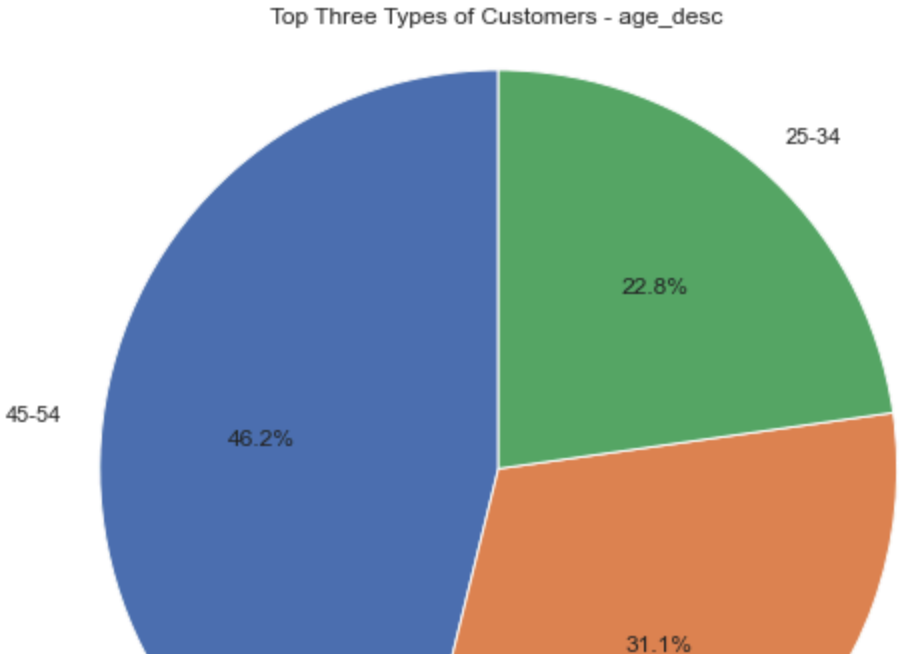

Bottom Three Types of Customers
1 ADULT KIDS
21.9%
SINGLE MALE
44.2%
34.0%
UNKNOWN

In [33]:
```
import matplotlib.pyplot as plt

# Calculate the percentage of each type of customer
customer_counts = demographic_data['hh_comp_desc'].value_counts(normalize
top_three_customers = customer_counts.head(3)
bottom_three_customers = customer_counts.tail(3)

# Define colors in the red family
red_family_colors = [(1.0, 0.4, 0.4, 0.6), (1.0, 0.25, 0.25, 0.6), (1.0,

# Pie chart for top three customers with red family colors
plt.figure(figsize=(8, 8))
plt.pie(top_three_customers, labels=top_three_customers.index, autopct='%
plt.title('Top Three Types of Customers')
plt.axis('equal')
plt.show()

# Pie chart for bottom three customers with red family colors
plt.figure(figsize=(8, 8))
plt.pie(bottom_three_customers, labels=bottom_three_customers.index, auto
plt.title('Bottom Three Types of Customers')
plt.axis('equal')
plt.show()
```

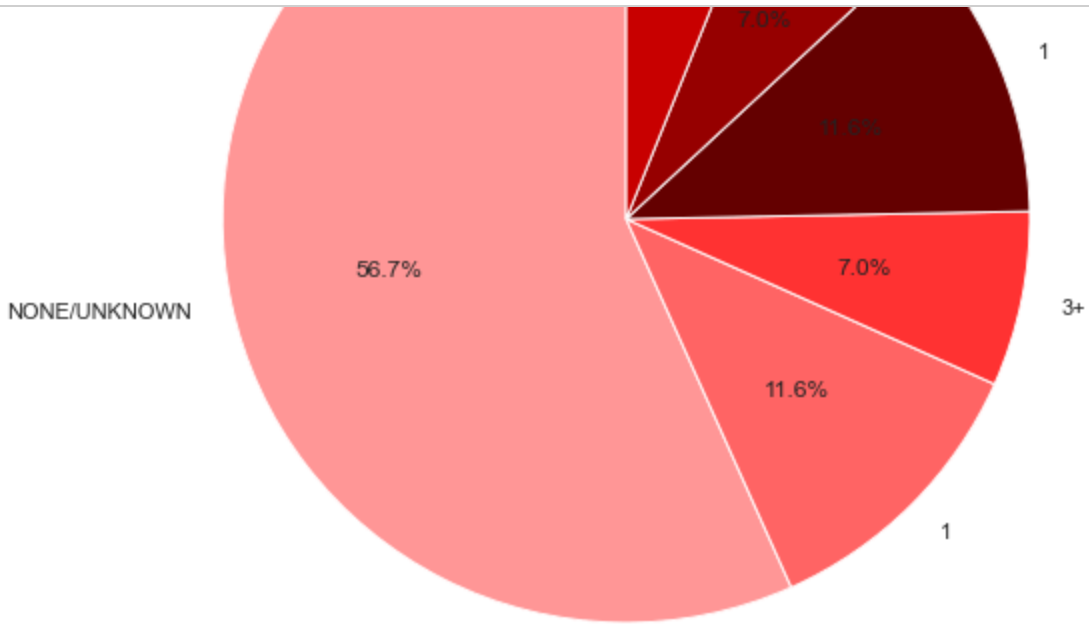



Bottom Three Types of Customers
1 ADULT KIDS
21.9%
SINGLE MALE
44.2%
34.0%
UNKNOWN

In [34]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data from CSV file
demographic_data = pd.read_csv('/Users/nisharams/Desktop/csv/hh_demograpl

# Define the demographic segments
segments = ['age_desc', 'marital_status_code', 'income_desc', 'homeowner_
            'hh_comp_desc', 'household_size_desc', 'kid_category_desc']

# Function to plot pie charts for top and bottom three types of customers
def plot_top_bottom_customers(segment):
    customer_counts = demographic_data[segment].value_counts(normalize=T
    top_three_customers = customer_counts.head(3)
    bottom_three_customers = customer_counts.tail(3)

    # Pie chart for top three customers
    plt.figure(figsize=(8, 8))
    plt.pie(top_three_customers, labels=top_three_customers.index, autop
    plt.title(f'Top Three Types of Customers - {segment}')
    plt.axis('equal')
    plt.show()

    # Pie chart for bottom three customers
    plt.figure(figsize=(8, 8))
    plt.pie(bottom_three_customers, labels=bottom_three_customers.index,
    plt.title(f'Bottom Three Types of Customers - {segment}')
    plt.axis('equal')
    plt.show()

# Plot pie charts for each demographic segment
for segment in segments:
    plot_top_bottom_customers(segment)
```



In [35]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load data from CSV file
demographic_data = pd.read_csv('/Users/nisharams/Desktop/csv/hh_demograpl

# Define the demographic segments
segments = ['age_desc', 'marital_status_code', 'income_desc', 'homeowner_
            'hh_comp_desc', 'household_size_desc', 'kid_category_desc']

# Function to plot pie chart for top and bottom three types of customers
def plot_top_bottom_customers(segment):
    customer_counts = demographic_data[segment].value_counts(normalize=T
    top_three_customers = customer_counts.head(3)
    bottom_three_customers = customer_counts.tail(3)

    # Combine top and bottom customer data
    combined_data = pd.concat([top_three_customers, bottom_three_custome

    # Colors for top and bottom customer segments
    colors = ['#FF9999', '#FF6666', '#FF3333', '#660000', '#990000', '#C

    # Pie chart for top and bottom customers
    plt.figure(figsize=(8, 8))
    plt.pie(combined_data, labels=combined_data.index, autopct='%1.1f%%'
    plt.title(f'Top and Bottom Types of Customers - {segment}')
    plt.axis('equal')
    plt.show()

# Plot pie charts for each demographic segment
for segment in segments:
    plot_top_bottom_customers(segment)
```



In [36]:
```
import pandas as pd

# Load data from CSV file
demographic_data = pd.read_csv('/Users/nisharams/Desktop/csv/hh_demograpl

# Define the demographic segments
segments = ['age_desc', 'marital_status_code', 'income_desc', 'homeowner_
            'hh_comp_desc', 'household_size_desc', 'kid_category_desc']

# Function to get the combined list of top 3 and bottom 3 customer types
def get_top_bottom_customers(segment):
    customer_counts = demographic_data[segment].value_counts(normalize=T
    top_three_customers = customer_counts.head(3)
    bottom_three_customers = customer_counts.tail(3)

    # Combine top and bottom customer data
    combined_data = pd.concat([top_three_customers, bottom_three_custome
    combined_data = combined_data.reset_index()
    combined_data.columns = ['Customer Type', 'Percentage']

    return combined_data

# Create a dictionary to store the combined data for each demographic seg
combined_data_dict = {}

# Get the combined data for each demographic segment
for segment in segments:
    combined_data = get_top_bottom_customers(segment)
    combined_data_dict[segment] = combined_data

# Print the combined data for each demographic segment
for segment in segments:
    print(f"\nTop and Bottom Customer Types - {segment}")
    print(combined_data_dict[segment])
```

```
Top and Bottom Customer Types - age_desc
  Customer Type  Percentage
0         45-54    0.359551
1         35-44    0.242197
2         25-34    0.177278
3           65+    0.089888
4         55-64    0.073658
5         19-24    0.057428

Top and Bottom Customer Types - marital_status_code
  Customer Type  Percentage
0             U    0.429463
1             A    0.424469
2             B    0.146067
3             U    0.429463
4             A    0.424469
5             B    0.146067

Top and Bottom Customer Types - income_desc
  Customer Type  Percentage
0        50-74K    0.239700
1        35-49K    0.214732
2        75-99K    0.119850
3         250K+    0.013733
4      175-199K    0.013733
5      200-249K    0.006242

Top and Bottom Customer Types - homeowner_desc
     Customer Type  Percentage
0        HOMEOWNER    0.629213
1          UNKNOWN    0.290886
2           RENTER    0.052434
3           RENTER    0.052434
4  PROBABLE RENTER    0.013733
5   PROBABLE OWNER    0.013733

Top and Bottom Customer Types - hh_comp_desc
      Customer Type  Percentage
0  2 ADULTS NO KIDS    0.318352
1     2 ADULTS KIDS    0.233458
2     SINGLE FEMALE    0.179775
3       SINGLE MALE    0.118602
4           UNKNOWN    0.091136
5      1 ADULT KIDS    0.058677

Top and Bottom Customer Types - household_size_desc
  Customer Type  Percentage
0             2    0.397004
1             1    0.318352
2             3    0.136080
3             3    0.136080
4            5+    0.082397
5             4    0.066167

Top and Bottom Customer Types - kid_category_desc
  Customer Type  Percentage
```

0	NONE/UNKNOWN	0.696629
1	1	0.142322
2	3+	0.086142
3	1	0.142322
4	3+	0.086142
5	2	0.074906

In [ ]: