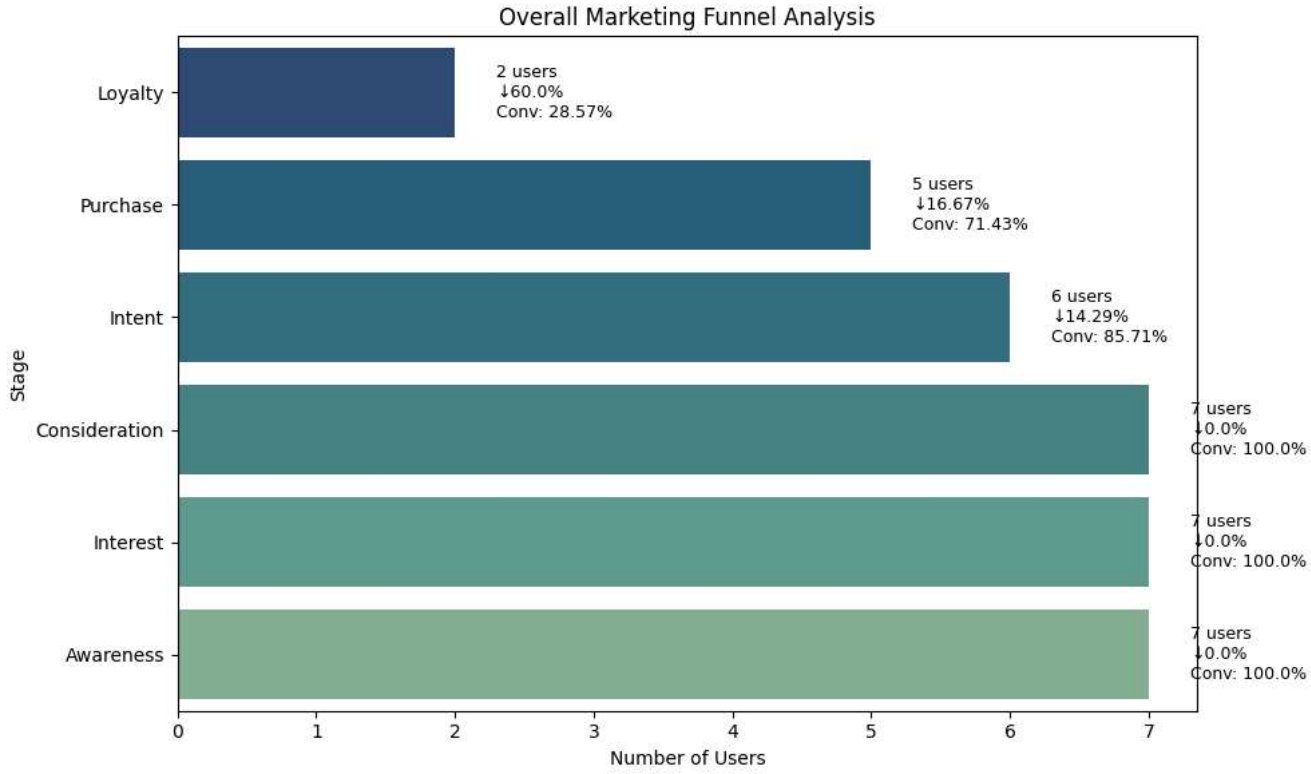
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Simulate Real-Like Data
data = [
    # Format: [CustomerID, Region, Channel, Stage]
    ["C001", "North", "Email", "Awareness"],
    ["C001", "North", "Email", "Interest"],
    ["C001", "North", "Email", "Consideration"],
    ["C001", "North", "Email", "Intent"],
    ["C001", "North", "Email", "Purchase"],
    ["C002", "South", "Social", "Awareness"],
    ["C002", "South", "Social", "Interest"],
    ["C002", "South", "Social", "Consideration"],
    ["C003", "West", "Organic", "Awareness"],
    ["C003", "West", "Organic", "Interest"],
    ["C003", "West", "Organic", "Consideration"],
    ["C003", "West", "Organic", "Intent"],
    ["C003", "West", "Organic", "Purchase"],
    ["C004", "East", "Paid", "Awareness"],
    ["C004", "East", "Paid", "Interest"],
    ["C004", "East", "Paid", "Consideration"],
    ["C004", "East", "Paid", "Intent"],
    ["C005", "North", "Email", "Awareness"],
    ["C005", "North", "Email", "Interest"],
    ["C005", "North", "Email", "Consideration"],
    ["C005", "North", "Email", "Intent"],
    ["C005", "North", "Email", "Purchase"],
    ["C005", "North", "Email", "Loyalty"],
    ["C006", "South", "Paid", "Awareness"],
    ["C006", "South", "Paid", "Interest"],
    ["C006", "South", "Paid", "Consideration"],
    ["C006", "South", "Paid", "Intent"],
    ["C006", "South", "Paid", "Purchase"],
    ["C007", "West", "Social", "Awareness"],
    ["C007", "West", "Social", "Interest"],
    ["C007", "West", "Social", "Consideration"],
    ["C007", "West", "Social", "Intent"],
    ["C007", "West", "Social", "Purchase"],
    ["C007", "West", "Social", "Loyalty"],
]

df = pd.DataFrame(data, columns=["CustomerID", "Region", "Channel", "Stage"])

# Step 2: Prepare Data
stage_order = ['Awareness', 'Interest', 'Consideration', 'Intent', 'Purchase', 'Loyalty']
df["Stage"] = pd.Categorical(df["Stage"], categories=stage_order, ordered=True)

# Step 3: Funnel Calculation
funnel_counts = df.drop_duplicates(subset=["CustomerID", "Stage"]).groupby("Stage")["CustomerID"].count().reindex(stage_order)
funnel_df = pd.DataFrame({
    "Stage": stage_order,
    "Users": funnel_counts.values
})
funnel_df["DropOff(%)"] = funnel_df["Users"].shift(1)
funnel_df["DropOff(%)"] = ((funnel_df["DropOff(%)"] - funnel_df["Users"]) / funnel_df["DropOff(%)"] * 100).round(2).fillna(0)
funnel_df["ConversionRate(%)"] = (funnel_df["Users"] / funnel_df["Users"].iloc[0] * 100).round(2)

# Step 4: Visualize Funnel
plt.figure(figsize=(10, 6))
sns.barplot(data=funnel_df, y="Stage", x="Users", palette="crest")
plt.title("Overall Marketing Funnel Analysis")
plt.xlabel("Number of Users")
plt.ylabel("Stage")

for index, row in funnel_df.iterrows():
    plt.text(row.Users + 0.3, index, f"{row.Users} users\n↓{row['DropOff(%)']}%\nConv: {row['ConversionRate(%)']}%", va='center', fontsize=9)

plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



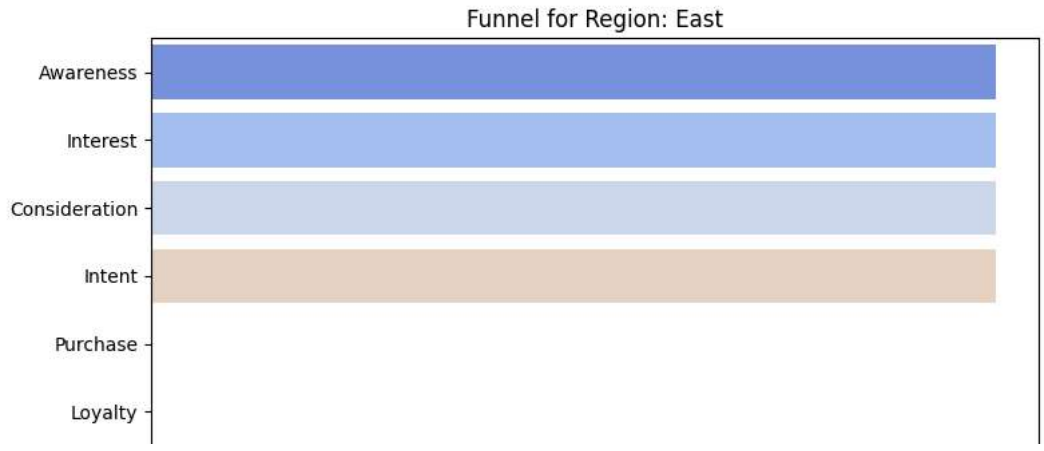
```
# Step 5: Show Summary
print("\n--- Funnel Summary ---")
print(funnel_df)
```

```
# Step 6: Segment Analysis Function
def segment_analysis(by):
    print(f"\n--- Funnel Breakdown by {by} ---")
    segment_stages = df.drop_duplicates(subset=["CustomerID", "Stage"]).groupby([by, "Stage"])["CustomerID"].count().unstack().fillna(0)
    print(segment_stages)

    # Optional: Segment Visuals
    for segment in segment_stages.index:
        values = segment_stages.loc[segment].reindex(stage_order).fillna(0)
        plt.figure(figsize=(8, 4))
        sns.barplot(x=values.values, y=stage_order, palette="coolwarm")
        plt.title(f"Funnel for {by}: {segment}")
        plt.xlabel("Users")
        plt.tight_layout()
        plt.show()

segment_analysis("Region")
segment_analysis("Channel")
```

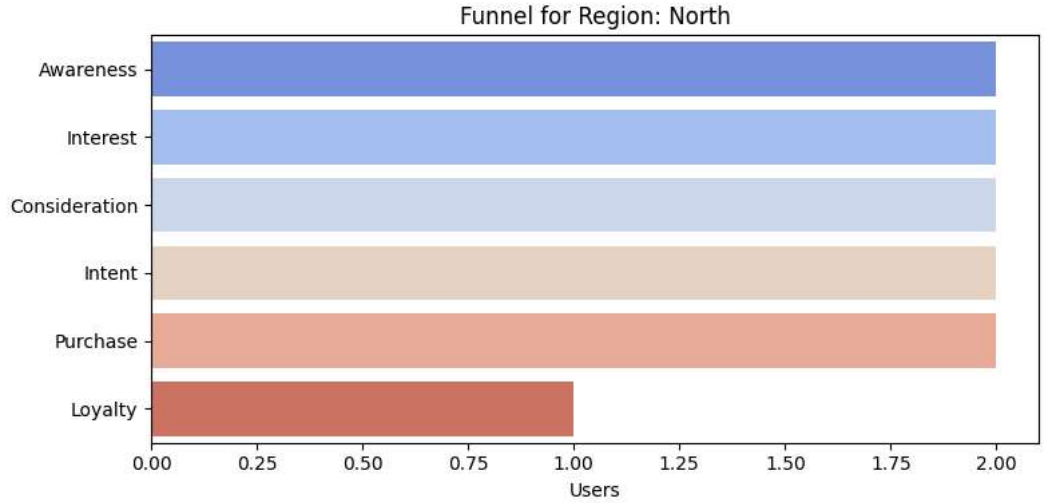
```
/tmp/ipython-input-1-3137643668.py:51: FutureWarning: The default of observed=False is deprecated and will be changed to True in a fu
  funnel_counts = df.drop_duplicates(subset=["CustomerID", "Stage"]).groupby("Stage")["CustomerID"].count().reindex(stage_order)
/tmp/ipython-input-1-3137643668.py:62: FutureWarning:

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

  sns.barplot(data=funnel_df, y="Stage", x="Users", palette="crest")
```



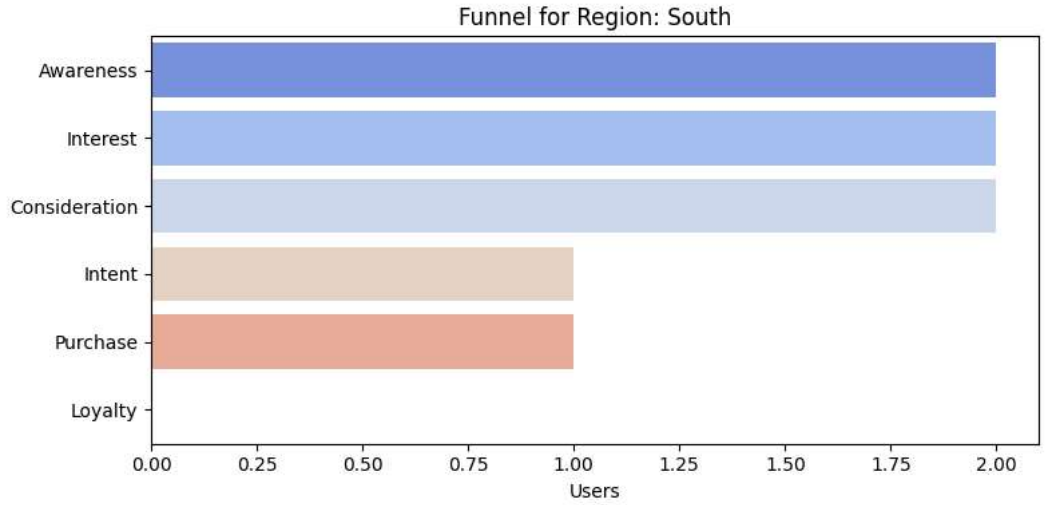

```
--- Funnel Summary ---
           Stage  Users  DropOff(%)  ConversionRate(%)
0      Awareness      7        0.00             100.00
1       Interest      7        0.00             100.00
2  Consideration      7        0.00             100.00
3         Intent      6       14.29              85.71
4       Purchase      5       16.67              71.43
5        Loyalty      2       60.00              28.57

--- Funnel Breakdown by Region ---
Stage   Awareness  Interest  Consideration  Intent  Purchase  Loyalty
Region
East            1         1              1       1         0        0
North           2         2              2       2         2        1
South           2         2              2       1         1        0
West            2         2              2       2         2        1
/tmp/ipython-input-1-3137643668.py:81: FutureWarning: The default of observed=False is deprecated and will be changed to True in a fu
  segment_stages = df.drop_duplicates(subset=["CustomerID", "Stage"]).groupby([by, "Stage"])["CustomerID"].count().unstack().fillna(0
/tmp/ipython-input-1-3137643668.py:88: FutureWarning:

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

  sns.barplot(x=values.values, y=stage_order, palette="coolwarm")
```



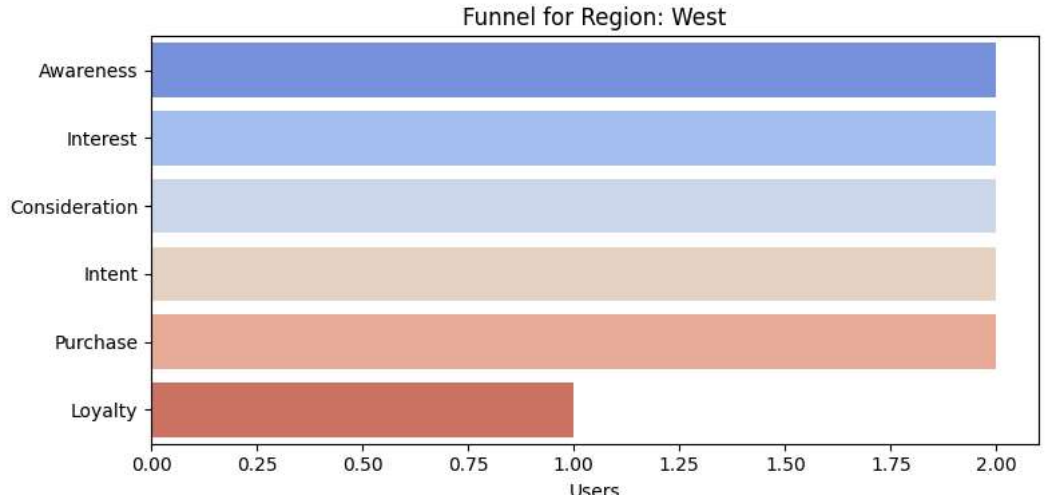

```
/tmp/ipython-input-1-3137643668.py:88: FutureWarning:

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

  sns.barplot(x=values.values, y=stage_order, palette="coolwarm")
```



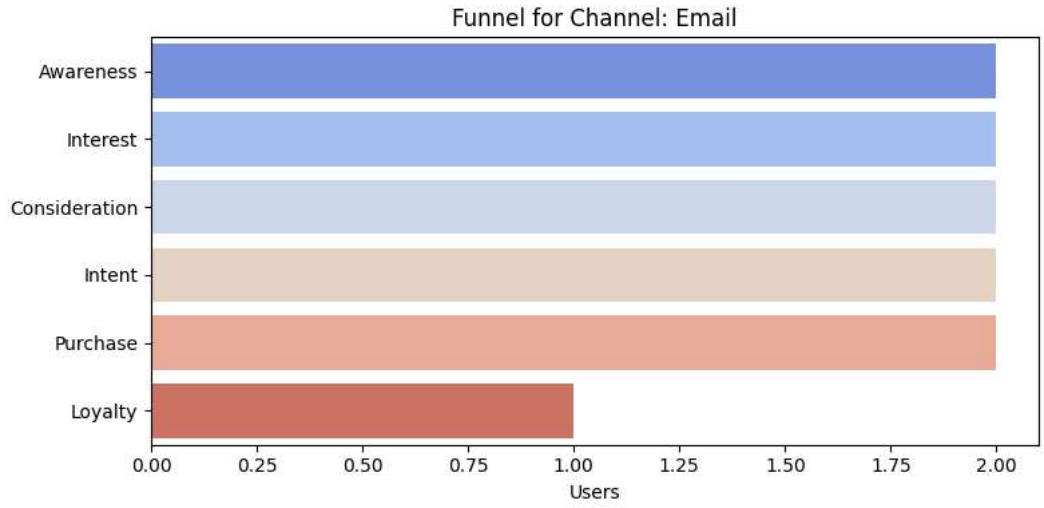

```
/tmp/ipython-input-1-3137643668.py:88: FutureWarning:

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

  sns.barplot(x=values.values, y=stage_order, palette="coolwarm")
```



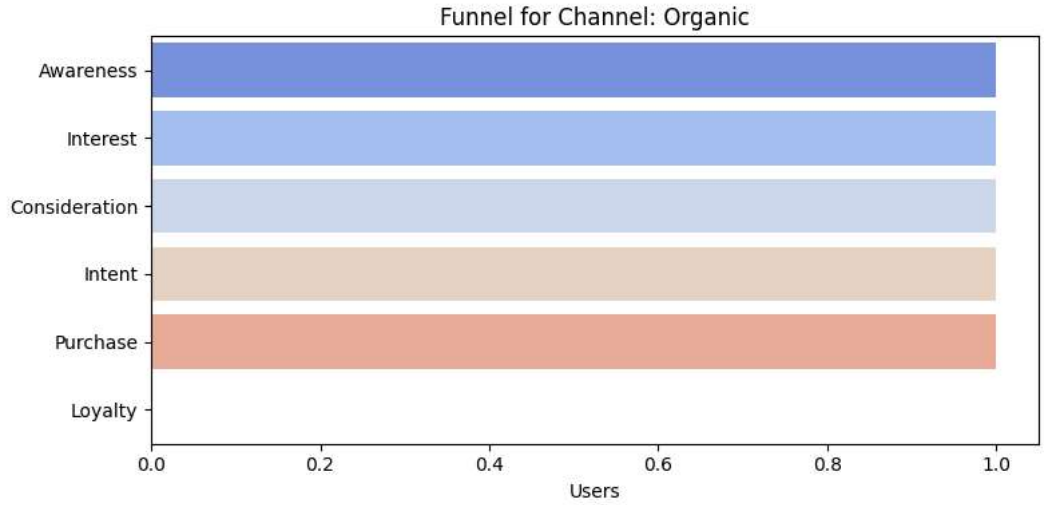

```
/tmp/ipython-input-1-3137643668.py:88: FutureWarning:

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

  sns.barplot(x=values.values, y=stage_order, palette="coolwarm")
```



```
/tmp/ipython-input-1-3137643668.py:81: FutureWarning: The default of observed=False is deprecated and will be changed to True in a fu
  segment_stages = df.drop_duplicates(subset=["CustomerID", "Stage"]).groupby([by, "Stage"])["CustomerID"].count().unstack().fillna(0
/tmp/ipython-input-1-3137643668.py:88: FutureWarning:

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

  sns.barplot(x=values.values, y=stage_order, palette="coolwarm")

--- Funnel Breakdown by Channel ---
Stage    Awareness  Interest  Consideration  Intent  Purchase  Loyalty
Channel
Email            2         2              2       2         2        1
Organic          1         1              1       1         1        0
Paid             2         2              2       2         1        0
Social           2         2              2       1         1        1
```




```
/tmp/ipython-input-1-3137643668.py:88: FutureWarning:

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

  sns.barplot(x=values.values, y=stage_order, palette="coolwarm")
```




```
/tmp/ipython-input-1-3137643668.py:88: FutureWarning:

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

  sns.barplot(x=values.values, y=stage_order, palette="coolwarm")
```