In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = r"C:\Users\chemi\Downloads\PROJECT-Strategies to Combat Shopping Car
df = pd.read_csv(file_path)

# 1. Count unique users
unique_users = df["User_ID"].nunique()
print(f"Unique Users: {unique_users}")

# 2. Count distinct user locations with names
distinct_locations = df["User_Location"].nunique()
location_names = df["User_Location"].unique()
print(f"Distinct User Locations: {distinct_locations}")
print("Location Names:", location_names)

# 3. Average, Minimum, and Maximum Cart Value (USD)
average_cart_value = df["Cart_Value"].mean()
min_cart_value = df["Cart_Value"].min()
max_cart_value = df["Cart_Value"].max()
print(f"Average Cart Value: ${average_cart_value:.2f}")
print(f"Minimum Cart Value: ${min_cart_value:.2f}")
print(f"Maximum Cart Value: ${max_cart_value:.2f}")

# 4. Total cart value
total_cart_value = df["Cart_Value"].sum()
print(f"Total Cart Value: ${total_cart_value:.2f}")

# Function to add data labels
def add_labels(ax, values, vertical=False):
    for p, val in zip(ax.patches, values):
        if vertical:
            ax.annotate(f'{val} ({round(p.get_height(), 3)}%)',
                        (p.get_x() + p.get_width() / 2., p.get_height() / 2.),
                        ha='center', va='center', fontsize=10, color='white', fo
        else:
            ax.annotate(f'{val} ({round(p.get_height(), 3)}%)',
                        (p.get_x() + p.get_width() / 2., p.get_height() / 2.),
                        ha='center', va='center', fontsize=10, color='white', fo

# 5. Gender Distribution
gender_counts = df["Gender"].value_counts()
gender_percentage = df["Gender"].value_counts(normalize=True) * 100
print("Gender Distribution:")
print(pd.DataFrame({'Count': gender_counts, 'Percentage': gender_percentage}))
ax = gender_percentage.plot(kind='bar', color=['blue', 'pink'])
plt.title("Gender Distribution")
plt.ylabel("Percentage")
plt.xlabel("Gender")
add_labels(ax, gender_counts.values)
plt.show()

# 6. Most Common Abandonment Reasons
abandonment_counts = df["Abandonment_Reason"].value_counts()
abandonment_percentage = df["Abandonment_Reason"].value_counts(normalize=True) *
print("Most Common Abandonment Reasons:")
```

```
print(pd.DataFrame({'Count': abandonment_counts, 'Percentage': abandonment_perce
ax = abandonment_percentage.plot(kind='bar', color='red')
plt.title("Most Common Abandonment Reasons")
plt.ylabel("Percentage")
plt.xlabel("Reason")
add_labels(ax, abandonment_counts.values, vertical=True)
plt.show()

# 7. Cart Status Distribution
cart_status_counts = df["Cart_Status"].value_counts()
cart_status_percentage = df["Cart_Status"].value_counts(normalize=True) * 100
print("Cart Status Distribution:")
print(pd.DataFrame({'Count': cart_status_counts, 'Percentage': cart_status_perce
ax = cart_status_percentage.plot(kind='bar', color='green')
plt.title("Cart Status Distribution")
plt.ylabel("Percentage")
plt.xlabel("Status")
add_labels(ax, cart_status_counts.values)
plt.show()

# 8. Average Session Duration
average_session_duration = df["Session_Duration"].mean()
print(f"Average Session Duration: {average_session_duration:.2f} seconds")

print(df["Session_Duration"].describe())


# Function to add data labels
def add_labels(ax, values, vertical=False):
    for p, val in zip(ax.patches, values):
        if vertical:
            ax.annotate(f'{val} ({round(p.get_height(), 3)}%)',
                        (p.get_x() + p.get_width() / 2., p.get_height() / 2.),
                        ha='center', va='center', fontsize=10, color='white', fo
        else:
            ax.annotate(f'{val} ({round(p.get_height(), 3)}%)',
                        (p.get_x() + p.get_width() / 2., p.get_height() / 2.),
                        ha='center', va='center', fontsize=10, color='white', fo



# 9. Top Referral Mediums
referral_counts = df["Referral_Medium"].value_counts()
referral_percentage = df["Referral_Medium"].value_counts(normalize=True) * 100
print("Top Referral Mediums:")
print(pd.DataFrame({'Count': referral_counts, 'Percentage': referral_percentage}
ax = referral_percentage.plot(kind='bar', color='purple')
plt.title("Top Referral Mediums")
plt.ylabel("Percentage")
plt.xlabel("Referral Medium")
add_labels(ax, referral_counts.values, vertical=True)
plt.show()

# 10. Device Type Analysis
device_counts = df["Device_Type"].value_counts()
device_percentage = df["Device_Type"].value_counts(normalize=True) * 100
print("Device Type Analysis:")
print(pd.DataFrame({'Count': device_counts, 'Percentage': device_percentage}))
ax = device_percentage.plot(kind='bar', color='orange')
plt.title("Device Type Analysis")
```

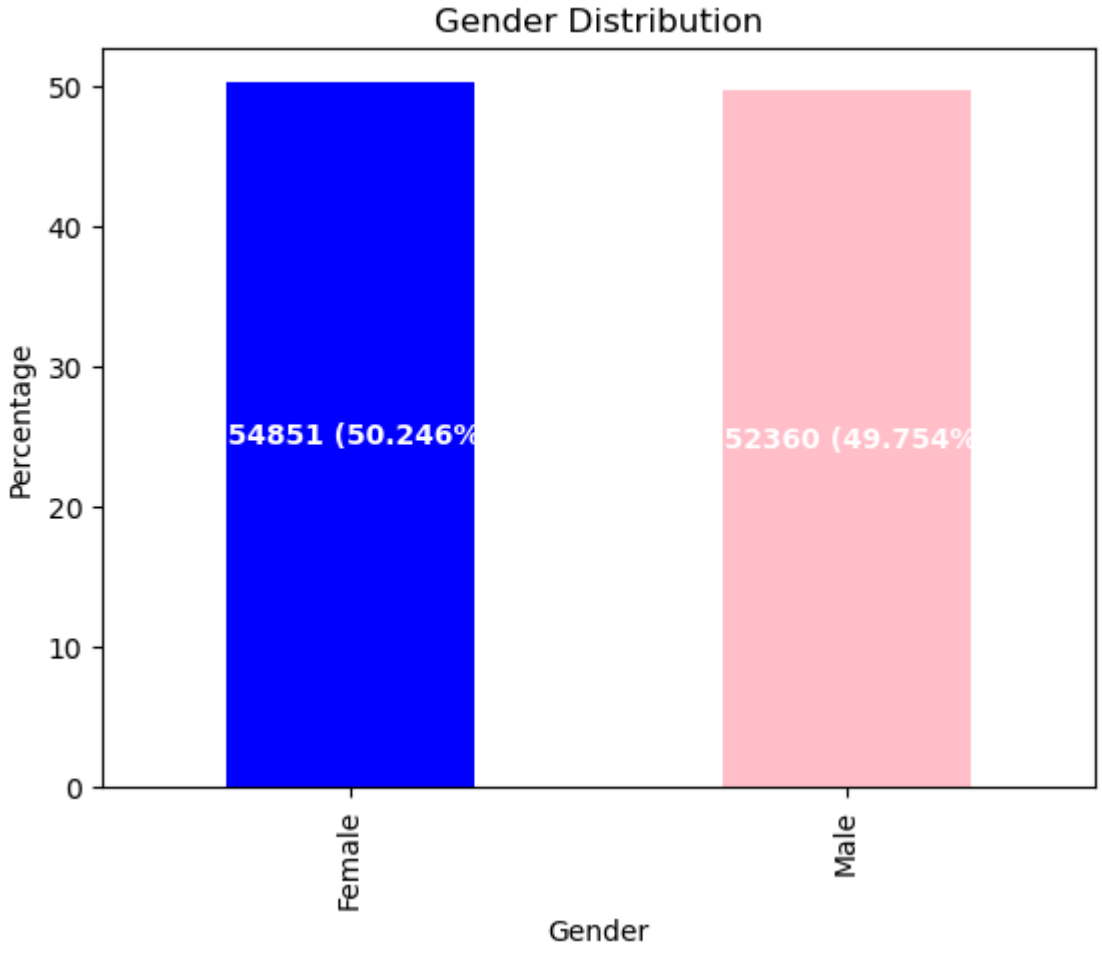
```
plt.ylabel("Percentage")
plt.xlabel("Device Type")
add_labels(ax, device_counts.values, vertical=True)
plt.show()



# 11. Month-wise Cart Paid and Abandoned Rate and Value


# Convert Session_Date to datetime format
df["Session_Date"] = pd.to_datetime(df["Session_Date"])
df["Session_Month"] = df["Session_Date"].dt.month
month_order = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
month_labels = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "



monthly_status = df.groupby(["Session_Month", "Cart_Status"]).agg({"Cart_Value":
monthly_pivot = monthly_status.pivot(index="Session_Month", columns="Cart_Status
monthly_pivot["Abandonment_Rate"] = (monthly_pivot["Abandoned"] / (monthly_pivot

print("Month-wise Cart Paid and Abandoned Rate:")
print(monthly_pivot)



monthly_status = df.groupby(["Session_Month", "Cart_Status"]).agg({"Cart_Value":
monthly_pivot = monthly_status.pivot(index="Session_Month", columns="Cart_Status
monthly_pivot["Total"] = monthly_pivot.sum(axis=1)
monthly_pivot["Paid_Rate"] = (monthly_pivot.get("Paid", 0) / monthly_pivot["Tota
monthly_pivot["Abandonment_Rate"] = (monthly_pivot.get("Abandoned", 0) / monthly
monthly_pivot = monthly_pivot.reindex(month_order)
monthly_pivot.index = month_labels

# Plot stacked column chart
fig, ax1 = plt.subplots(figsize=(12, 6))
monthly_pivot[["Paid", "Abandoned"]].plot(kind='bar', stacked=True, ax=ax1, colo
ax1.set_ylabel("Count of Users")
ax1.set_xlabel("Month")
ax1.set_title("Month-wise Cart Paid and Abandoned Count")
ax1.legend(title="Cart Status")

# Add data labels
for p in ax1.patches:
    if p.get_height() > 0:
        ax1.annotate(f'{int(p.get_height())}',
                     (p.get_x() + p.get_width() / 2., p.get_y() + p.get_height()
                     ha='center', va='center', fontsize=10, color='white', fontw

# Plot line chart for abandonment and paid rates
ax2 = ax1.twinx()
ax2.plot(monthly_pivot.index, monthly_pivot["Abandonment_Rate"], color='red', ma
ax2.plot(monthly_pivot.index, monthly_pivot["Paid_Rate"], color='green', marker=
ax2.set_ylabel("Percentage (%)")
ax2.legend(loc='upper right')

# Add data labels to lines
for i, txt in enumerate(monthly_pivot["Abandonment_Rate"]):
    ax2.annotate(f'{round(txt, 1)}%', (i, txt), ha='center', va='bottom', fontsi
```

```
for i, txt in enumerate(monthly_pivot["Paid_Rate"]):
    ax2.annotate(f'{round(txt, 1)}%', (i, txt), ha='center', va='top', fontsize=

plt.show()


# 12. Total Potential Revenue Lost Due to Abandonment
total_lost_revenue = df[df["Cart_Status"] == "Abandoned"]["Cart_Value"].sum()
print(f"Total Potential Revenue Lost Due to Abandonment: ${total_lost_revenue:,.

# 13. Percentage of Possible Revenue Lost Due to Abandonment
total_revenue = df["Cart_Value"].sum()
percentage_lost_revenue = (total_lost_revenue / total_revenue) * 100
print(f"Percentage of Possible Revenue Lost Due to Abandonment: {percentage_lost

# 14. Average Cart Value for Abandoned vs. Purchased Carts
avg_cart_value = df.groupby("Cart_Status")["Cart_Value"].mean()
print("Average Cart Value:")
print(avg_cart_value)

plt.figure(figsize=(6, 4))
sns.barplot(x=avg_cart_value.index, y=avg_cart_value.values, palette=["red", "gr
plt.title("Average Cart Value for Abandoned vs. Purchased Carts")
plt.ylabel("Average Cart Value (USD)")
plt.xlabel("Cart Status")
for i, val in enumerate(avg_cart_value.values):
    plt.text(i, val, f'${val:.2f}', ha='center', va='bottom', fontsize=10, fontw
plt.show()


# 15. Total Cart Value Paid and Abandoned
total_cart_value = df.groupby("Cart_Status")["Cart_Value"].sum()
total_cart_value_percentage = (total_cart_value / total_revenue) * 100
print("Total Cart Value for Paid and Abandoned Carts:")
print(total_cart_value)

plt.figure(figsize=(6, 4))
ax = sns.barplot(x=total_cart_value.index, y=total_cart_value.values, palette=["
plt.title("Total Cart Value: Paid vs. Abandoned")
plt.ylabel("Total Cart Value (USD)")
plt.xlabel("Cart Status")
for p, perc in zip(ax.patches, total_cart_value_percentage.values):
    ax.annotate(f'${p.get_height():,.3f}\n({perc:.3f}%)', (p.get_x() + p.get_wid
                ha='center', va='center', fontsize=10, fontweight='bold', color=
plt.show()




# 16. Revenue Loss by Purchase Category
revenue_loss_by_category = df[df["Cart_Status"] == "Abandoned"].groupby("Purchas
print("Revenue Loss by Purchase Category:")
print(revenue_loss_by_category)

plt.figure(figsize=(10, 5))
ax = revenue_loss_by_category.sort_values(ascending=False).plot(kind='bar', colo
plt.title("Revenue Loss by Purchase Category")
plt.ylabel("Lost Revenue (USD)")
plt.xlabel("Purchase Category")
for p in ax.patches:
```

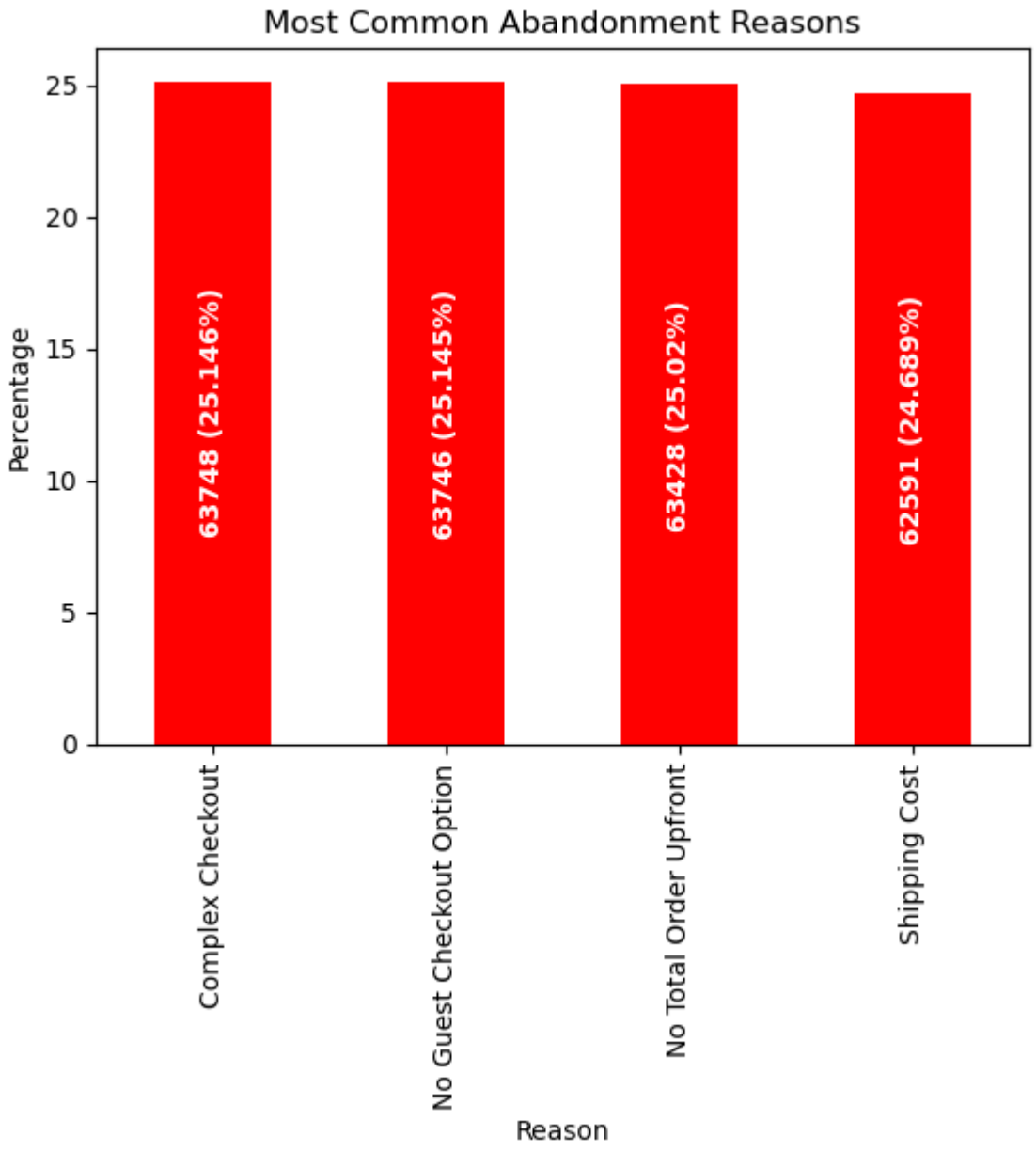
```
    ax.annotate(f'${p.get_height():,.2f}', (p.get_x() + p.get_width() / 2., p.ge
                ha='center', va='center', fontsize=10, fontweight='bold', rotati
plt.show()



# 17. Device Type with Abandonment Rate
device_abandonment = df[df["Cart_Status"] == "Abandoned"].groupby("Device_Type")
device_total = df.groupby("Device_Type").size()
device_abandonment_rate = (device_abandonment / device_total) * 100

device_abandonment_rate = device_abandonment_rate.sort_values(ascending=False).r
plt.figure(figsize=(8, 8))
plt.pie(device_abandonment_rate, labels=device_abandonment_rate.index, autopct='
plt.title("Device Type Abandonment Rate")
plt.show()

# 18. Referral Sources with Abandonment Rate
referral_abandonment = df[df["Cart_Status"] == "Abandoned"].groupby("Referral_Me
referral_total = df.groupby("Referral_Medium").size()
referral_abandonment_rate = (referral_abandonment / referral_total) * 100

referral_abandonment_rate = referral_abandonment_rate.sort_values(ascending=Fals
plt.figure(figsize=(8, 8))
plt.pie(referral_abandonment_rate, labels=referral_abandonment_rate.index, autop
plt.title("Referral Source Abandonment Rate")
plt.show()

# 19. Month-wise Cart Abandonment
month_abandonment = df[df["Cart_Status"] == "Abandoned"].groupby("Session_Month"
month_total = df.groupby("Session_Month").size()
month_abandonment_rate = (month_abandonment / month_total) * 100

plt.figure(figsize=(8, 5))
ax = sns.lineplot(x=month_labels, y=month_abandonment_rate.values, marker='o', c
plt.title("Month-wise Cart Abandonment Rate")
plt.ylabel("Abandonment Rate (%)")
plt.xlabel("Month")
for i, rate in enumerate(month_abandonment_rate.values):
    plt.text(i, rate, f'{rate:.2f}%', ha='center', va='bottom', fontsize=10, fon
plt.show()



# 20. Overall Conversion Rate Analysis
total_sessions = len(df)
purchased_sessions = len(df[df["Cart_Status"] == "Paid"])
conversion_rate = (purchased_sessions / total_sessions) * 100
print(f"Overall Conversion Rate: {conversion_rate:.3f}%")

# 21. Conversion Rate by Device Type
device_conversion = df[df["Cart_Status"] == "Paid"].groupby("Device_Type").size(
device_total = df.groupby("Device_Type").size()
device_conversion_rate = (device_conversion / device_total) * 100

plt.figure(figsize=(8, 8))
plt.pie(device_conversion_rate, labels=device_conversion_rate.index, autopct='%1
plt.title("Conversion Rate by Device Type")
plt.show()
```

```
# 22. Conversion Rate by Referral Source
referral_conversion = df[df["Cart_Status"] == "Paid"].groupby("Referral_Medium")
referral_total = df.groupby("Referral_Medium").size()
referral_conversion_rate = (referral_conversion / referral_total) * 100

plt.figure(figsize=(8, 8))
plt.pie(referral_conversion_rate, labels=referral_conversion_rate.index, autopct
plt.title("Conversion Rate by Referral Source")
plt.show()

# 23. Impact of Session Duration on Abandonment
session_bins = [0, 20, 50, 80, 120]
df["Session_Bin"] = pd.cut(df["Session_Duration"], bins=session_bins, labels=["0
abandonment_rate = df[df["Cart_Status"] == "Abandoned"].groupby("Session_Bin").s

plt.figure(figsize=(8, 5))
ax = sns.lineplot(x=abandonment_rate.index, y=abandonment_rate.values, marker='o
plt.title("Impact of Session Duration on Abandonment")
plt.ylabel("Abandonment Rate (%)")
plt.xlabel("Session Duration")
for i, rate in enumerate(abandonment_rate.values):
    plt.text(i, rate, f'{rate:.3f}%', ha='center', va='bottom', fontsize=10, fon
plt.show()

# 24. Most Abandoned Product Categories
category_abandonment = df[df["Cart_Status"] == "Abandoned"].groupby("Purchase_Ca

plt.figure(figsize=(10, 6))
ax = sns.barplot(x=category_abandonment.index, y=category_abandonment.values, pa
plt.title("Most Abandoned Product Categories")
plt.ylabel("Number of Abandoned Carts")
plt.xlabel("Product Category")
plt.xticks(rotation=45)
for p, value in zip(ax.patches, category_abandonment.values):
    ax.annotate(f'{value}', (p.get_x() + p.get_width() / 2., p.get_height() / 2.
                ha='center', va='center', fontsize=10, fontweight='bold', rotati
plt.show()

# 25. Abandoned Users by Session Range
abandoned_sessions = df[df["Cart_Status"] == "Abandoned"].groupby("Session_Bin")

plt.figure(figsize=(8, 6))
ax = sns.barplot(x=abandoned_sessions.index, y=abandoned_sessions.values, palett
plt.title("Abandoned Users by Session Range")
plt.ylabel("Number of Abandoned Users")
plt.xlabel("Session Duration Range")
for p, value in zip(ax.patches, abandoned_sessions.values):
    ax.annotate(f'{value}', (p.get_x() + p.get_width() / 2., p.get_height() / 2.
                ha='center', va='center', fontsize=10, fontweight='bold', rotati
plt.show()
```

```
Unique Users: 507211
Distinct User Locations: 6
Location Names: ['Virginia' 'Florida' 'New York' 'Chicago' 'Texas' 'California']
Average Cart Value: $259.76
Minimum Cart Value: $20.04
Maximum Cart Value: $499.91
Total Cart Value: $131752082.89
Gender Distribution:
         Count  Percentage
Gender
Female  254851   50.245559
Male    252360   49.754441
```

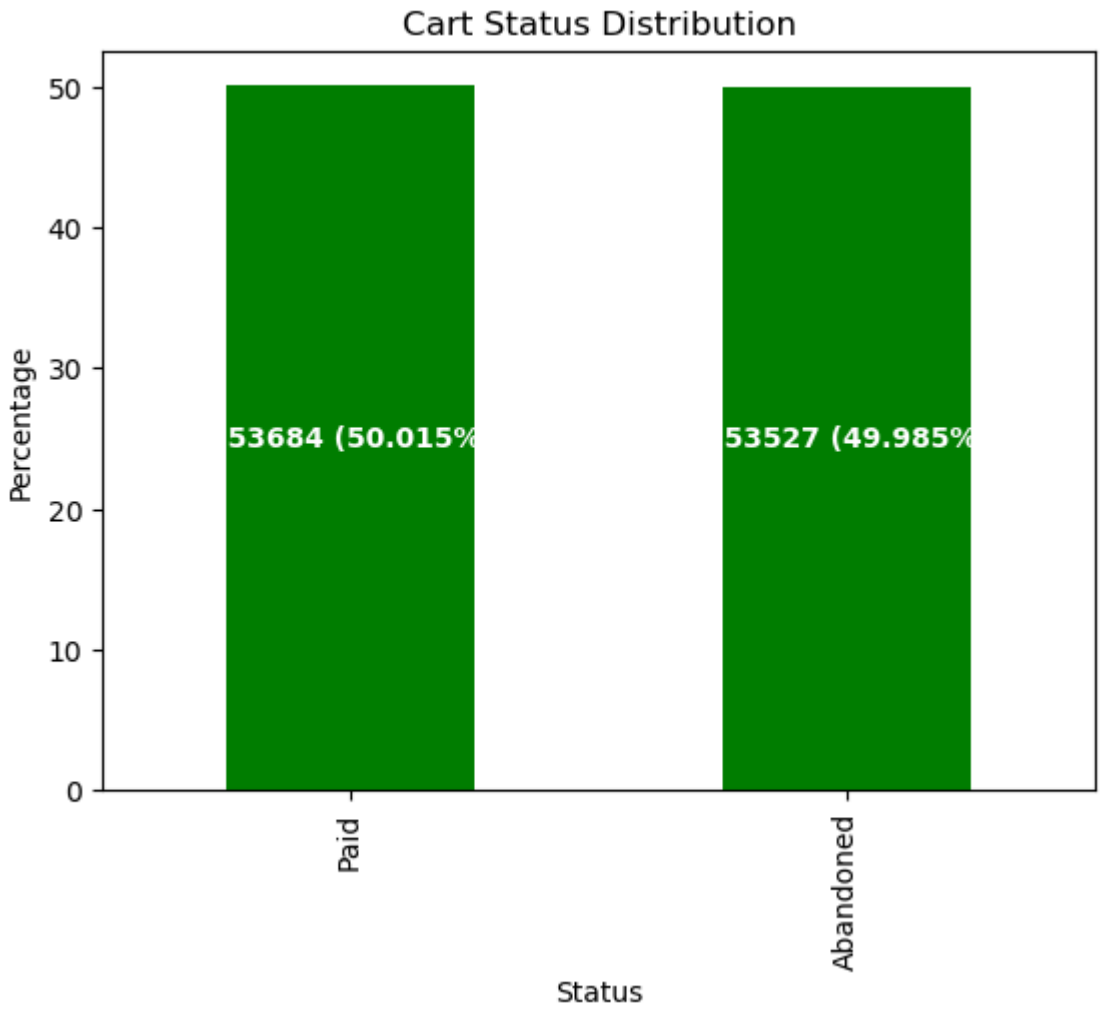



```
Most Common Abandonment Reasons:
                          Count  Percentage
Abandonment_Reason
Complex Checkout          63748   25.145851
No Guest Checkout Option  63746   25.145062
No Total Order Upfront    63428   25.019624
Shipping Cost             62591   24.689464
```

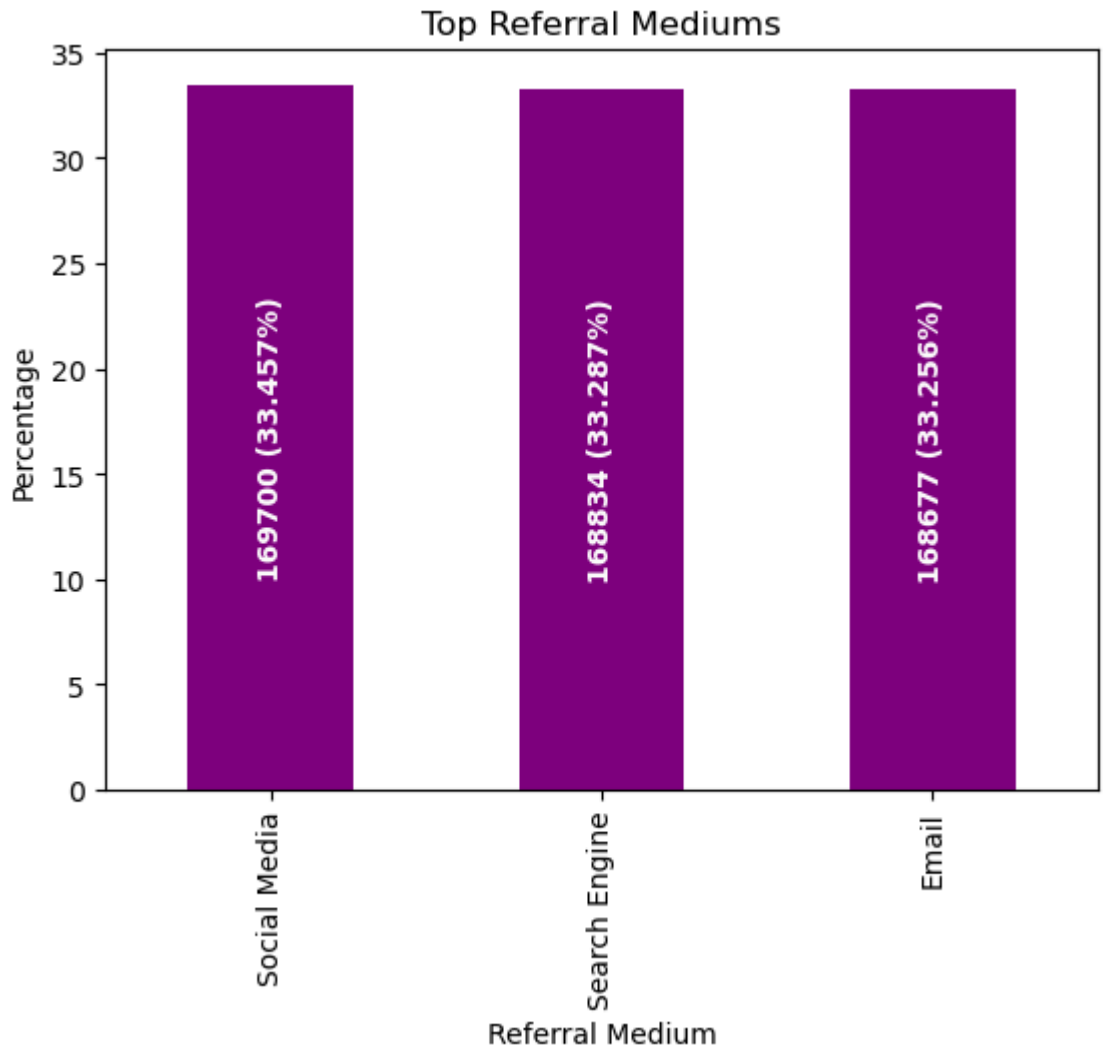


Cart Status Distribution:

Cart_Status	Count	Percentage
Paid	253684	50.015477
Abandoned	253527	49.984523

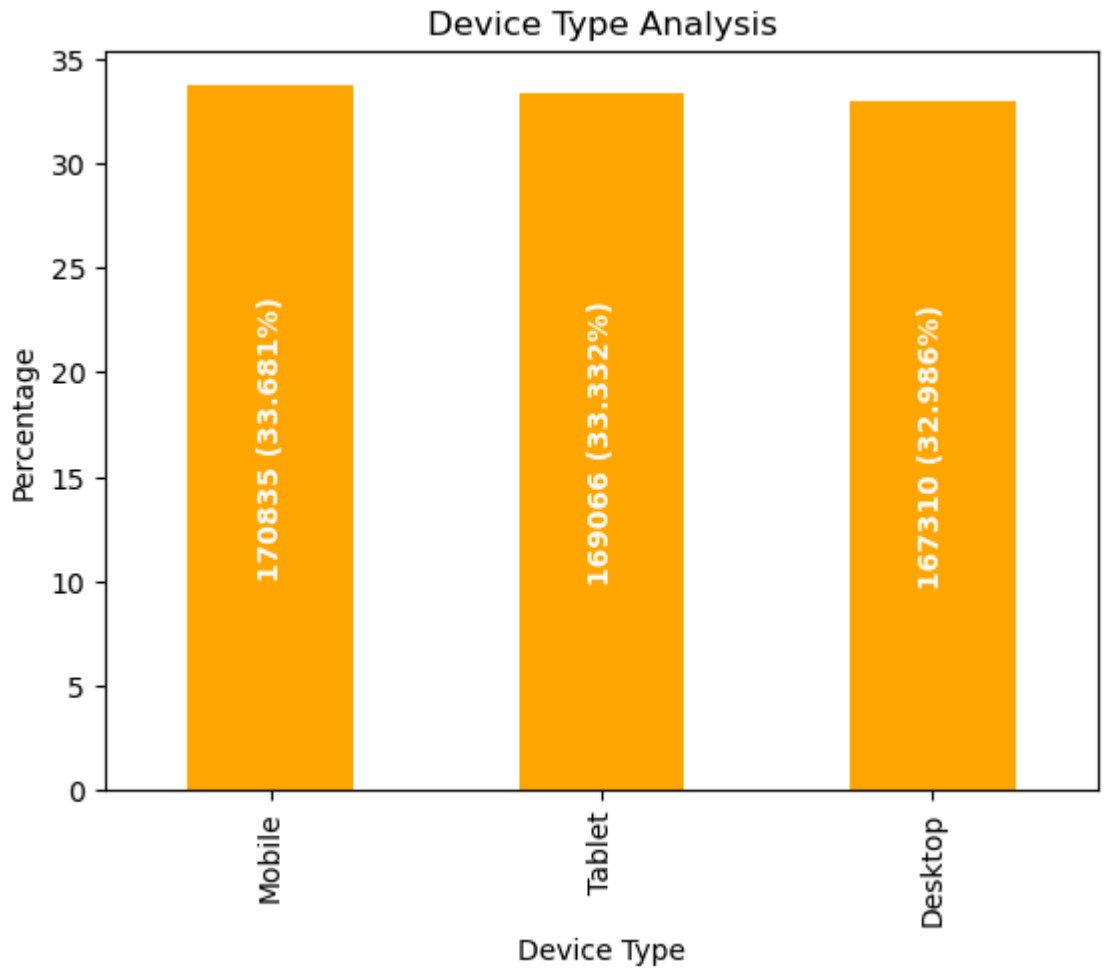



```
Average Session Duration: 62.39 seconds
count    507211.000000
mean         62.389885
std          33.465853
min           5.000000
25%          33.000000
50%          62.000000
75%          91.000000
max         120.000000
Name: Session_Duration, dtype: float64
Top Referral Mediums:
                 Count  Percentage
Referral_Medium
Social Media     169700   33.457476
Search Engine    168834   33.286739
Email            168677   33.255785
```



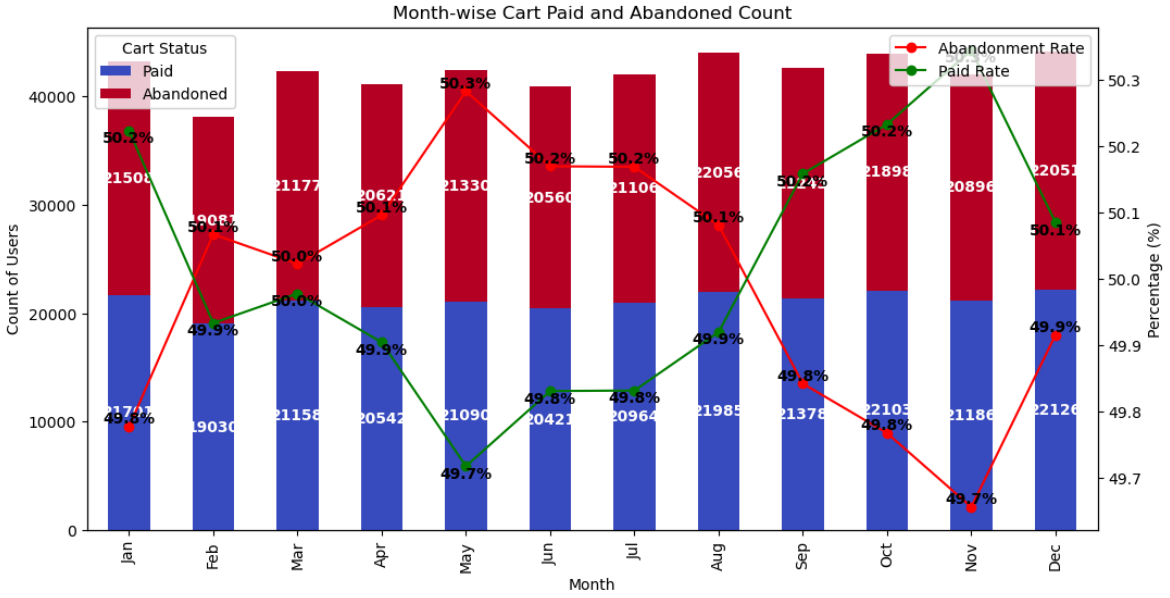
Device Type Analysis:

Device_Type	Count	Percentage
Mobile	170835	33.681249
Tablet	169066	33.332479
Desktop	167310	32.986272



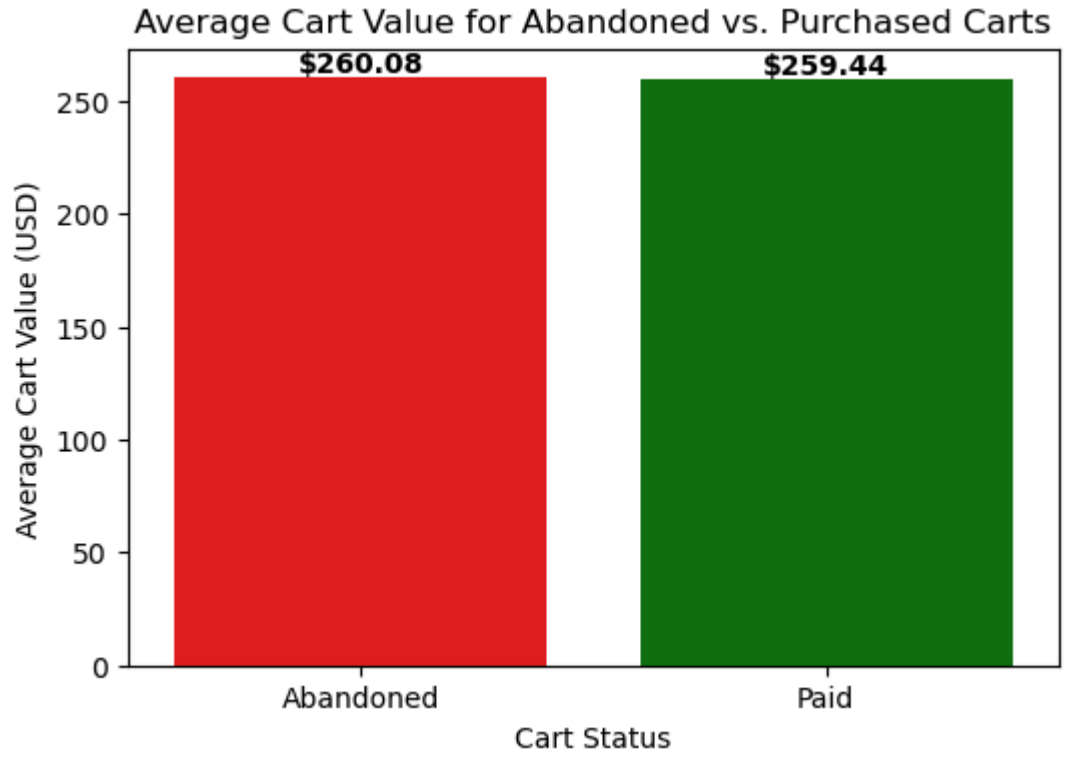
Month-wise Cart Paid and Abandoned Rate:

Session_Month	Abandoned	Paid	Abandonment_Rate
1	21508	21701	49.776667
2	19081	19030	50.066910
3	21177	21158	50.022440
4	20621	20542	50.095960
5	21330	21090	50.282885
6	20560	20421	50.169591
7	21106	20964	50.168766
8	22056	21985	50.080607
9	21243	21378	49.841627
10	21898	22103	49.767051
11	20896	21186	49.655435
12	22051	22126	49.915114



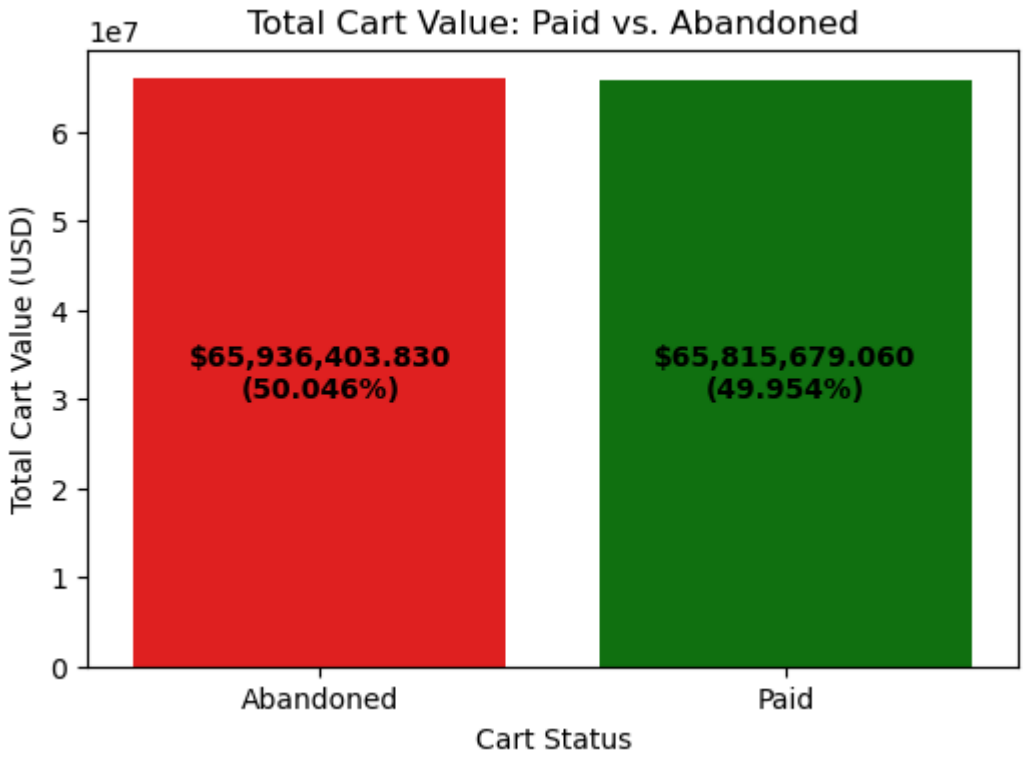

```
Total Potential Revenue Lost Due to Abandonment: $65,936,403.83
Percentage of Possible Revenue Lost Due to Abandonment: 50.05%
Average Cart Value:
Cart_Status
Abandoned    260.076457
Paid         259.439614
Name: Cart_Value, dtype: float64
```

```
C:\Users\chemi\AppData\Local\Temp\ipykernel_8476\1431955760.py:199: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=avg_cart_value.index, y=avg_cart_value.values, palette=["red", "g
reen"])
```

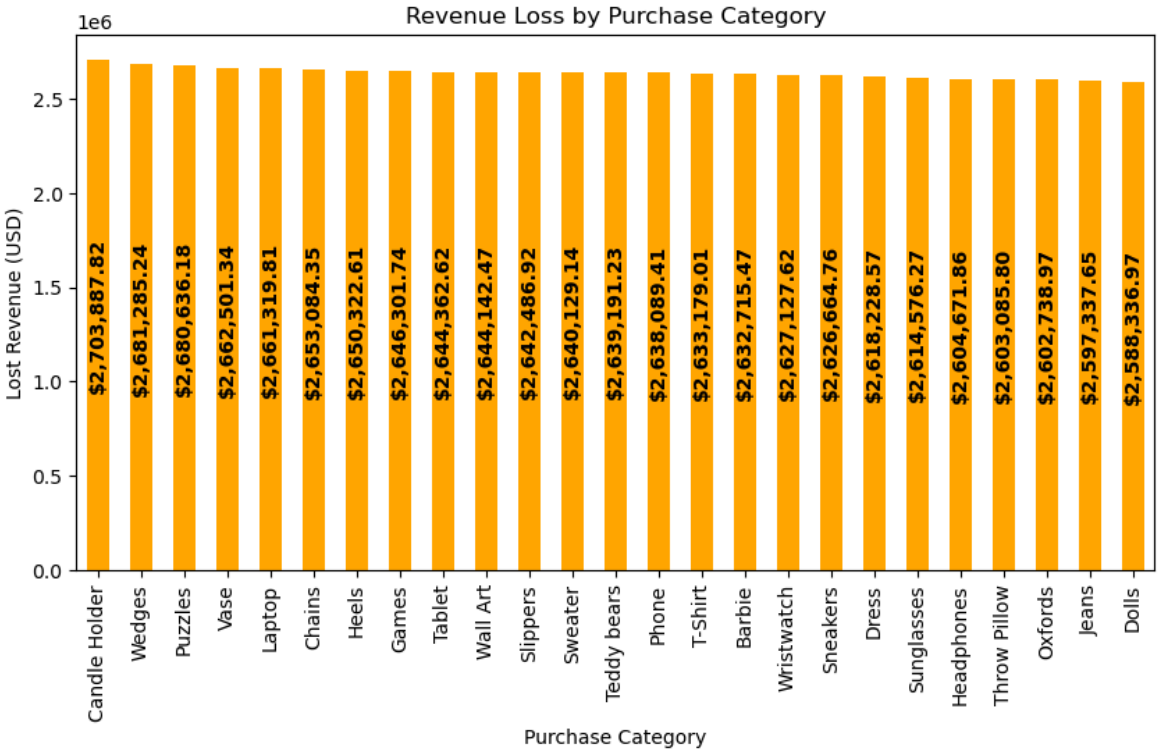


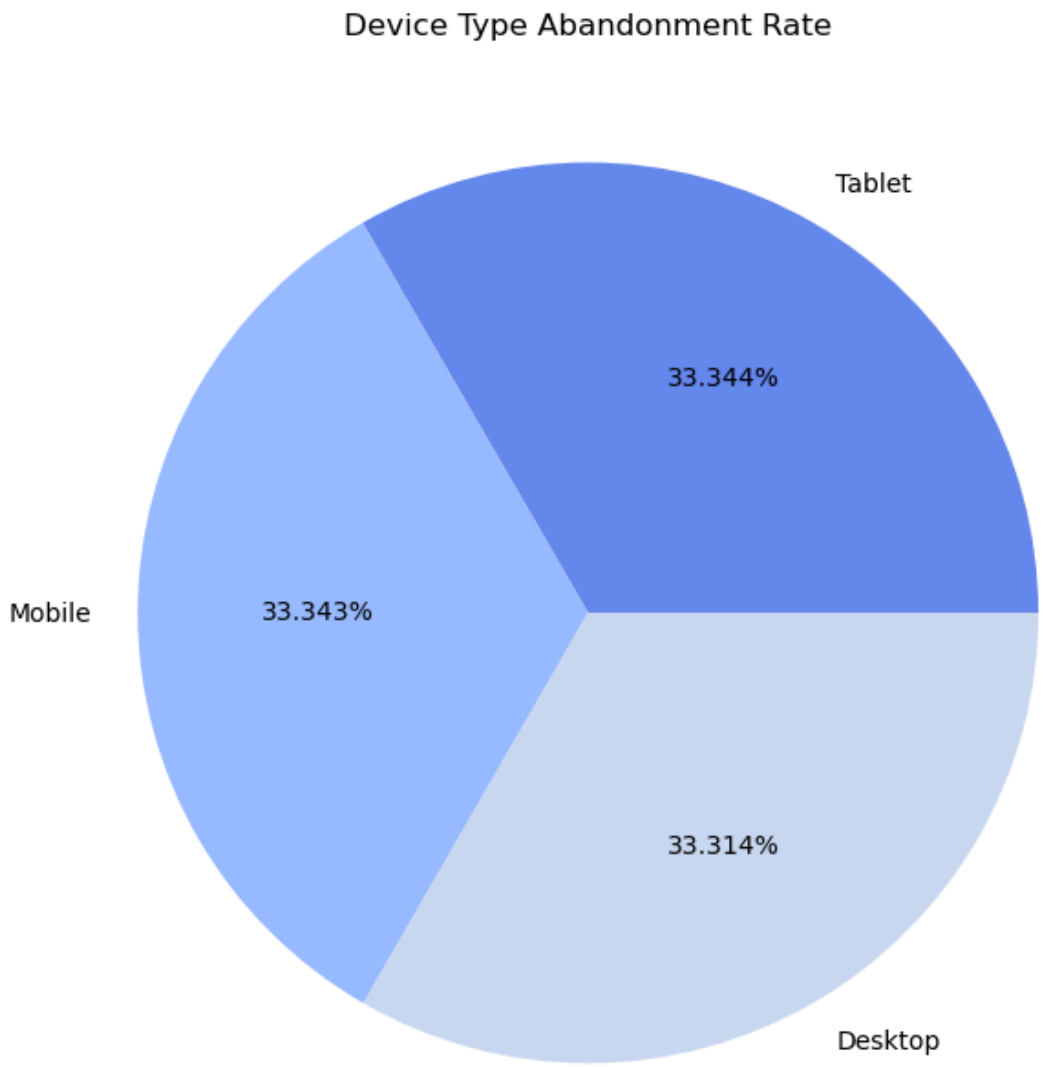
```
Total Cart Value for Paid and Abandoned Carts:
Cart_Status
Abandoned    65936403.83
Paid         65815679.06
Name: Cart_Value, dtype: float64
```

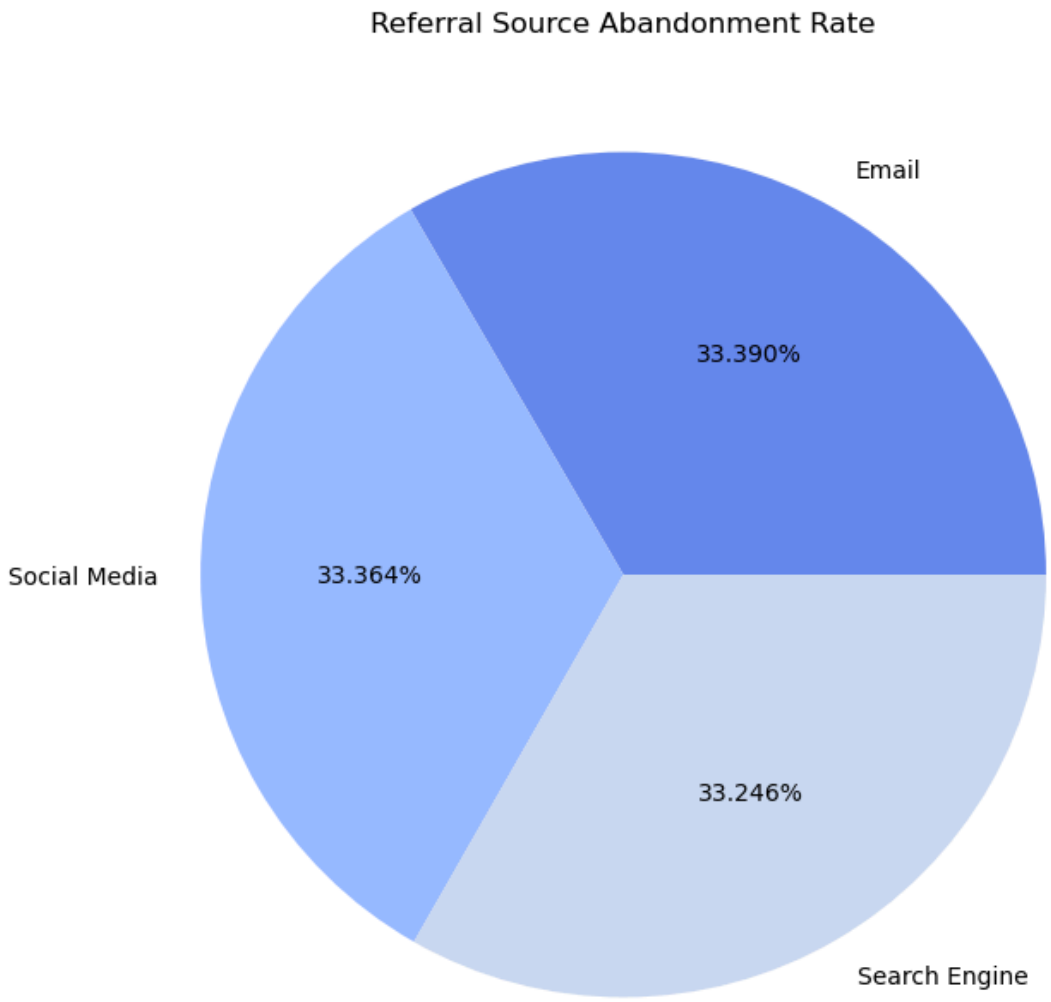
```
C:\Users\chemi\AppData\Local\Temp\ipykernel_8476\1431955760.py:215: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x=total_cart_value.index, y=total_cart_value.values, palette=
["red", "green"])
```

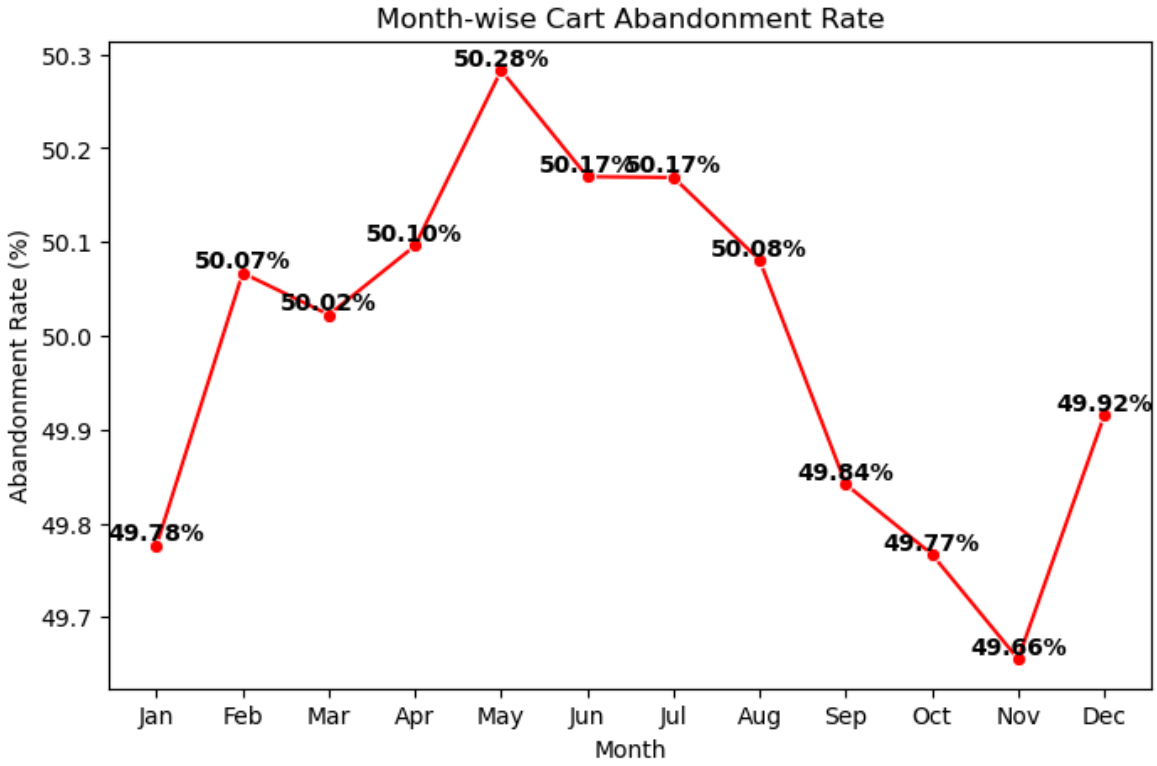


```
Revenue Loss by Purchase Category:
Purchase_Category
Barbie           2632715.47
Candle Holder    2703887.82
Chains           2653084.35
Dolls            2588336.97
Dress            2618228.57
Games            2646301.74
Headphones       2604671.86
Heels            2650322.61
Jeans            2597337.65
Laptop           2661319.81
Oxfords          2602738.97
Phone            2638089.41
Puzzles          2680636.18
Slippers         2642486.92
Sneakers         2626664.76
Sunglasses       2614576.27
Sweater          2640129.14
T-Shirt          2633179.01
Tablet           2644362.62
Teddy bears      2639191.23
Throw Pillow     2603085.80
Vase             2662501.34
Wall Art         2644142.47
Wedges           2681285.24
Wristwatch       2627127.62
Name: Cart_Value, dtype: float64
```

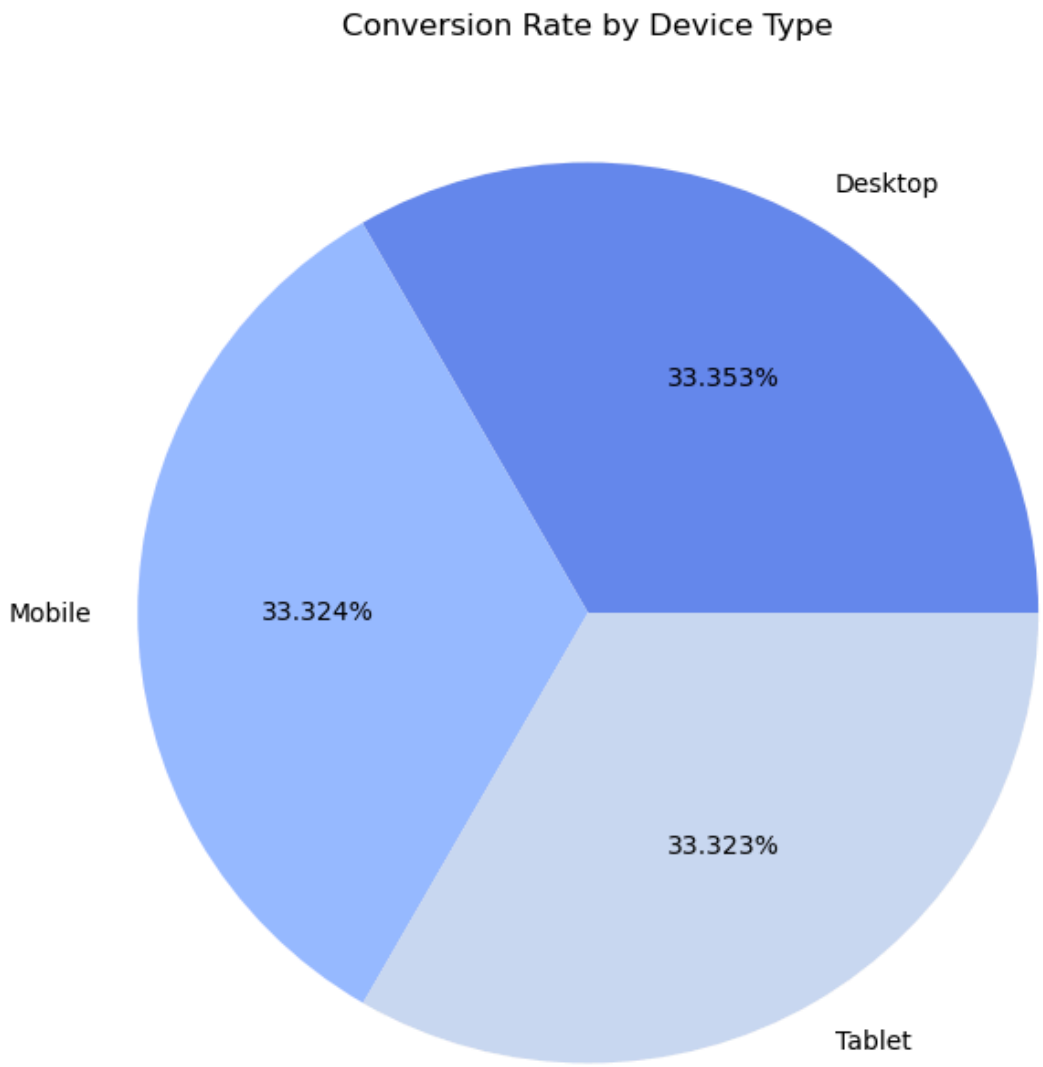


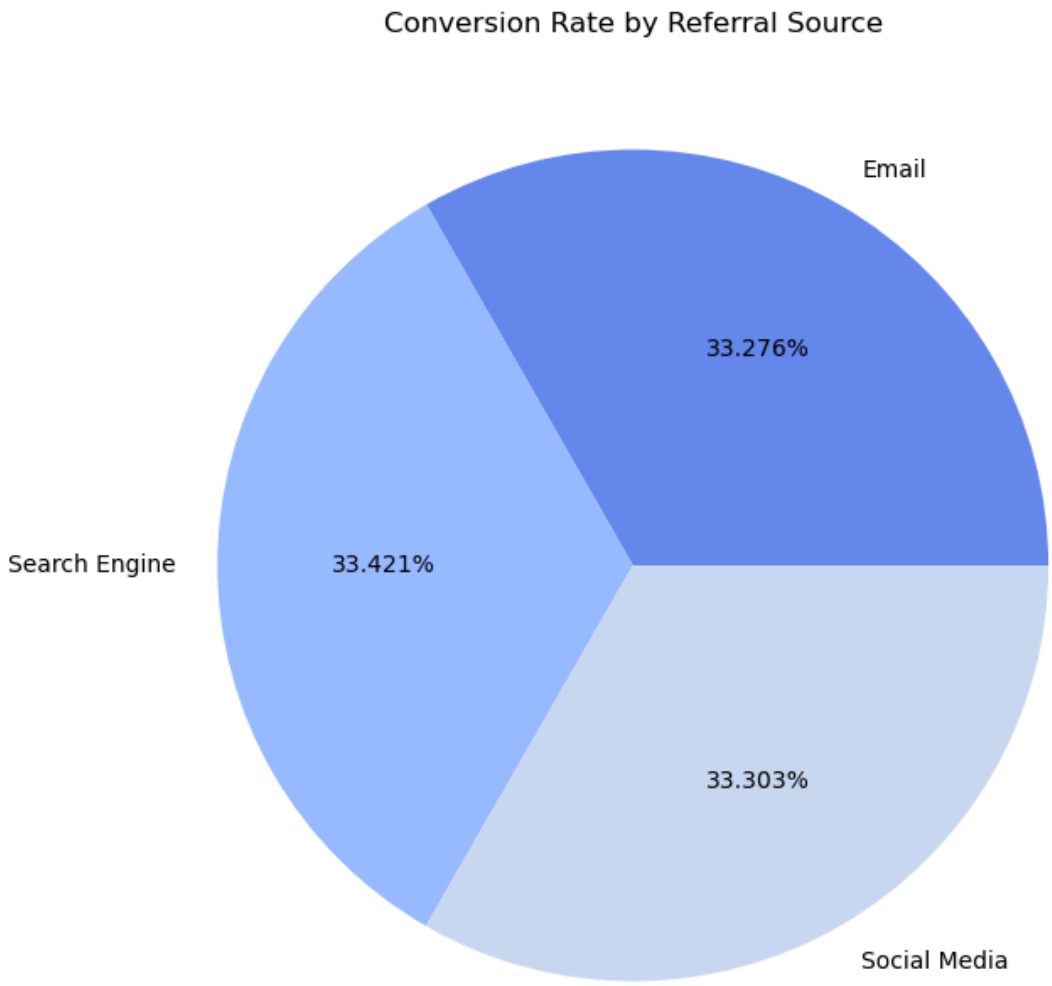

Device Type Abandonment Rate
Tablet
33.344%
Mobile
33.343%
33.314%
Desktop



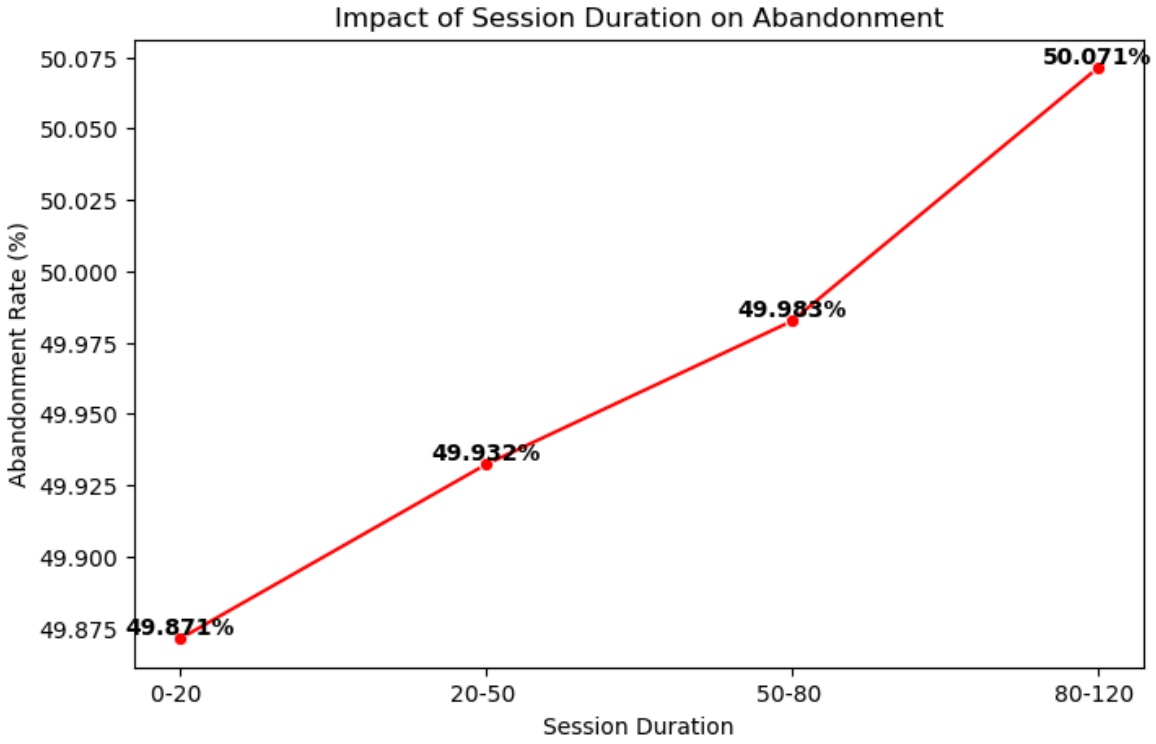



```
Overall Conversion Rate: 50.015%
```

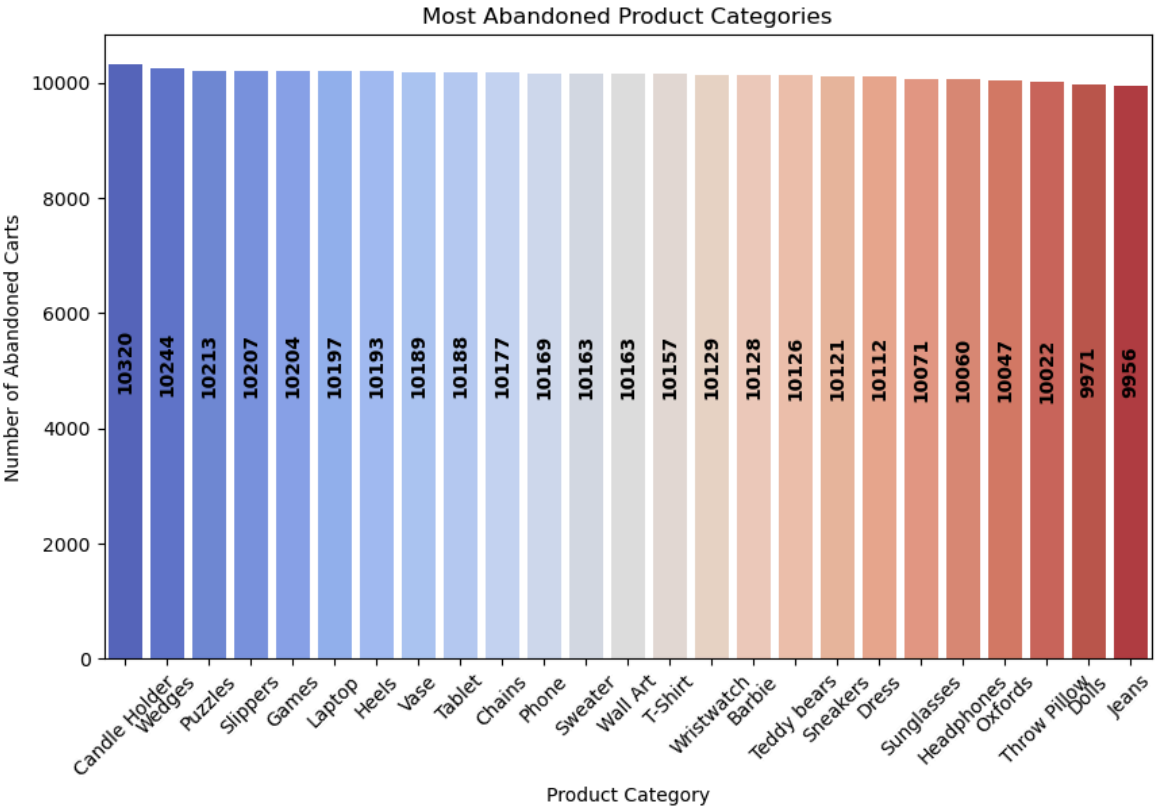

Conversion Rate by Device Type
Desktop
33.353%
Mobile
33.324%
33.323%
Tablet



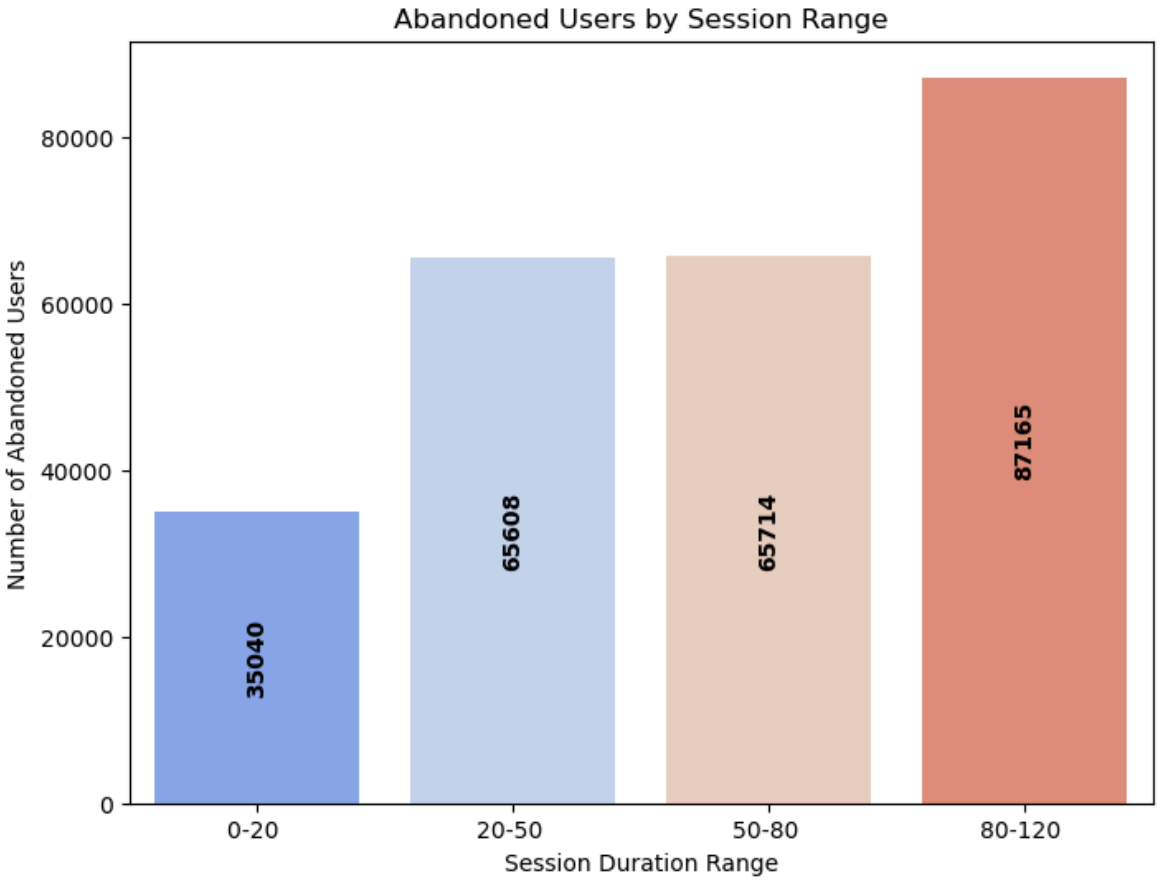

```
C:\Users\chemi\AppData\Local\Temp\ipykernel_8476\1431955760.py:311: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a f
uture version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  abandonment_rate = df[df["Cart_Status"] == "Abandoned"].groupby("Session_Bin").
size() / df.groupby("Session_Bin").size() * 100
C:\Users\chemi\AppData\Local\Temp\ipykernel_8476\1431955760.py:311: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a f
uture version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  abandonment_rate = df[df["Cart_Status"] == "Abandoned"].groupby("Session_Bin").
size() / df.groupby("Session_Bin").size() * 100
```




```
C:\Users\chemi\AppData\Local\Temp\ipykernel_8476\1431955760.py:326: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x=category_abandonment.index, y=category_abandonment.values, p
alette="coolwarm")
```



```
C:\Users\chemi\AppData\Local\Temp\ipykernel_8476\1431955760.py:337: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a f
uture version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  abandoned_sessions = df[df["Cart_Status"] == "Abandoned"].groupby("Session_Bi
n").size()
C:\Users\chemi\AppData\Local\Temp\ipykernel_8476\1431955760.py:340: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.barplot(x=abandoned_sessions.index, y=abandoned_sessions.values, palet
te="coolwarm")
```



In [ ]: