In [2]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\mishti\appdata\local\programs\python\python310\lib\site-packag
es (5.17.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\mishti\appdata\local\programs\python\python310\lib\si
te-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\mishti\appdata\local\programs\python\python310\lib\site-pac
kages (from plotly) (23.1)
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from matplotlib.ticker import FuncFormatter
%matplotlib inline
```

In [4]:
```
df = pd.read_csv(r"C:\Users\Mishti\OneDrive\Desktop\practice\practice1.csv")
```

In [5]:
```
df.head()
```

Out[5]:

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country
0	581482	12-09-2019	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom
1	581475	12-09-2019	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom
2	581475	12-09-2019	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom
3	581475	12-09-2019	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom
4	581475	12-09-2019	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 536350 entries, 0 to 536349
Data columns (total 8 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   TransactionNo  536350 non-null  object 
 1   Date           536350 non-null  object 
 2   ProductNo      536350 non-null  object 
 3   ProductName    536350 non-null  object 
 4   Price          536350 non-null  float64
 5   Quantity       536350 non-null  int64  
 6   CustomerNo     536295 non-null  float64
 7   Country        536350 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 32.7+ MB
```

In [7]:
```
df[df.duplicated()].head()
```

Out[7]:

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country
985	581497	12-09-2019	21481	Fawn Blue Hot Water Bottle	7.24	1	17497.0	United Kingdom
1365	581538	12-09-2019	23275	Set Of 3 Hanging Owls Ollie Beak	6.19	1	14446.0	United Kingdom
1401	581538	12-09-2019	22992	Revolver Wooden Ruler	6.19	1	14446.0	United Kingdom
1406	581538	12-09-2019	22694	Wicker Star	6.19	1	14446.0	United Kingdom
1409	581538	12-09-2019	23343	Jumbo Bag Vintage Christmas	6.19	1	14446.0	United Kingdom

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
TransactionNo     0
Date              0
ProductNo         0
ProductName       0
Price             0
Quantity          0
CustomerNo       55
Country           0
dtype: int64
```

In [9]:
```
df = df.dropna()
df.isnull().sum()
```

```
Out[9]:  TransactionNo    0
         Date             0
         ProductNo        0
         ProductName      0
         Price            0
         Quantity         0
         CustomerNo       0
         Country          0
         dtype: int64
```

```
In [12]: dateFormat = '%m/%d/%Y'
         df['Date'] = pd.to_datetime(df['Date'],format = 'mixed')
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 536295 entries, 0 to 536349
Data columns (total 8 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   TransactionNo  536295 non-null  object
 1   Date           536295 non-null  datetime64[ns]
 2   ProductNo      536295 non-null  object
 3   ProductName    536295 non-null  object
 4   Price          536295 non-null  float64
 5   Quantity       536295 non-null  int64
 6   CustomerNo     536295 non-null  float64
 7   Country        536295 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 36.8+ MB
```

```
In [13]: df.describe()
```

Out[13]:

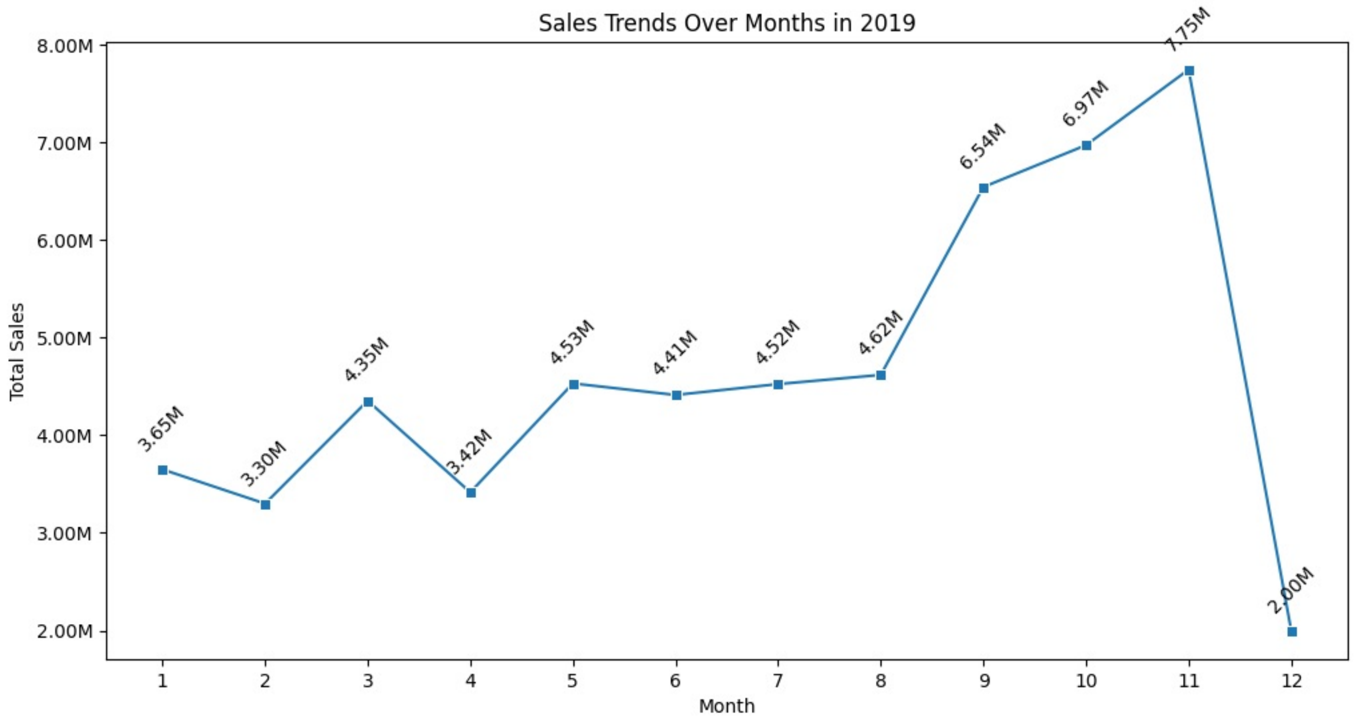
	Date	Price	Quantity	CustomerNo
count	536295	536295.000000	536295.000000	536295.000000
mean	2019-07-04 02:58:08.535227392	12.662031	9.923902	15227.893178
min	2018-12-01 00:00:00	5.130000	-80995.000000	12004.000000
25%	2019-03-28 00:00:00	10.990000	1.000000	13807.000000
50%	2019-07-20 00:00:00	11.940000	3.000000	15152.000000
75%	2019-10-19 00:00:00	14.090000	10.000000	16729.000000
max	2019-12-09 00:00:00	660.620000	80995.000000	18287.000000
std	NaN	8.490638	216.671641	1716.582932

```
In [14]: df = df.rename(columns={'TransactionNo':'Id_Transaction','ProductNo':'Id_Product','ProductName':'Product','Cust
         df.head()
```

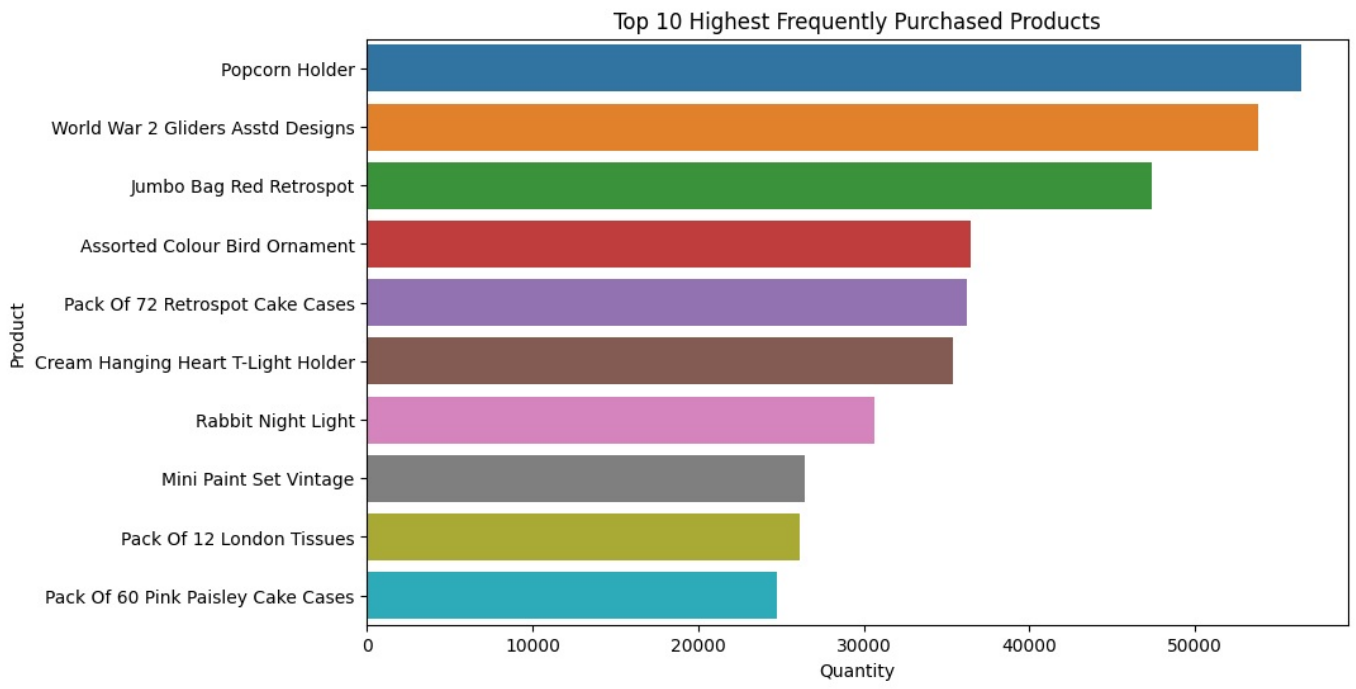
Out[14]:

	Id_Transaction	Date	Id_Product	Product	Price	Quantity	Id_Customer	Country
0	581482	2019-12-09	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom
1	581475	2019-12-09	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom
2	581475	2019-12-09	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom
3	581475	2019-12-09	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom
4	581475	2019-12-09	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom

```
In [15]: # Add a Total_Sales column by multiplying the values in the Quantity and Price columns
         df['Total_Sales'] = df['Quantity'] * df['Price']

         # Add Year and Month columns to observe data trends each month
         df['Year'] = df['Date'].dt.year
         df['Month'] = df['Date'].dt.month

         # Select data for the year 2019 to analyze the total sales trends monthly
         total_sales = df[df['Year']==2019].groupby('Month')['Total_Sales'].sum().reset_index()

         def format_millions_y(x, pos):
             return f'{x/1e6:.2f}M'

         # Visualization
         plt.figure(figsize=(12, 6))
         ax = sns.lineplot(x='Month', y='Total_Sales', data=total_sales, marker='s')
         ax.yaxis.set_major_formatter(FuncFormatter(format_millions_y))
         for index, row in total_sales.iterrows():
             plt.annotate(f'{row["Total_Sales"]/1e6:.2f}M', (row['Month'], row['Total_Sales']), textcoords="offset point
         plt.xlabel('Month')
         plt.ylabel('Total Sales')
         plt.title('Sales Trends Over Months in 2019')
         plt.xticks(range(1, 13))
         plt.show()
```

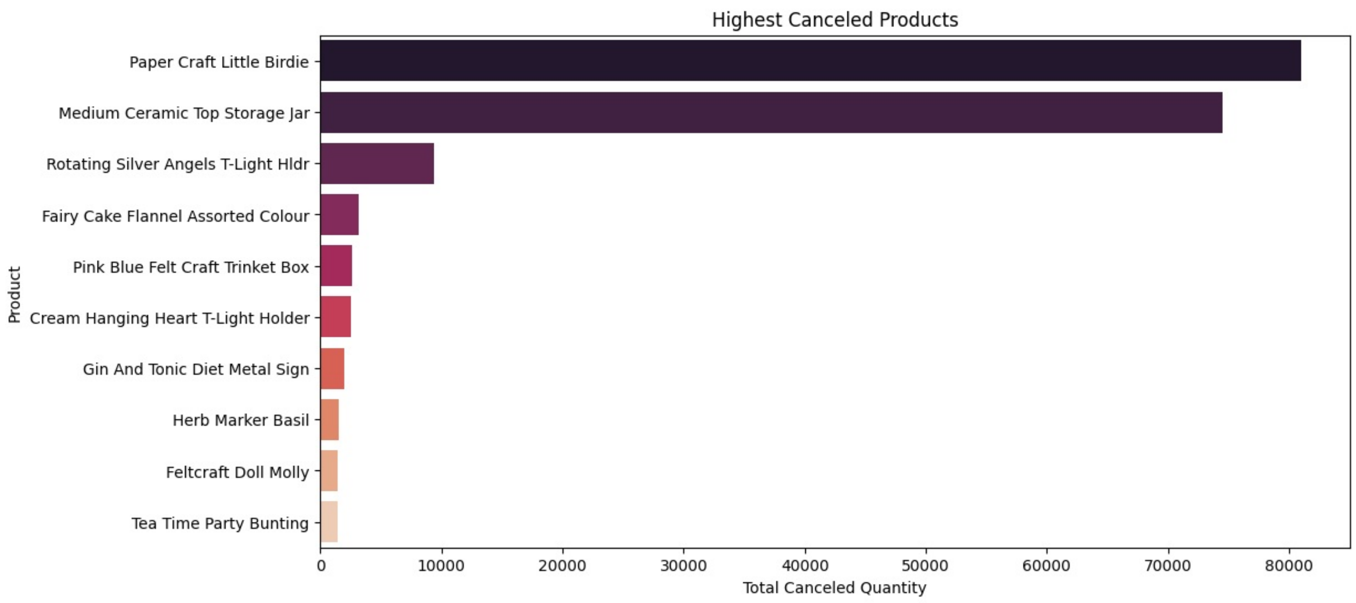


In [16]:
```
top_purchased_product = df.groupby(['Product'])['Quantity'].sum().reset_index()

# and then sort them in descending order for top 10 highest frequently purchased products
top_purchased_product = top_purchased_product.sort_values(by=['Quantity'], ascending = False)

# Visualization
plt.figure(figsize=(10, 6))
sns.barplot(x='Quantity', y='Product', data=top_purchased_product.head(10), palette='tab10')
plt.xlabel('Quantity')
plt.ylabel('Product')
plt.title('Top 10 Highest Frequently Purchased Products')
plt.show()
```



In [17]:
```
canceled_products = df[df['Quantity'] < 0].copy()
canceled_products.loc[:,'Quantity'] = abs(canceled_products['Quantity'])
total_canceled_quantity_per_product = canceled_products.groupby('Product')['Quantity'].sum().reset_index()

# and then sort them in descending order
total_canceled_quantity_per_product = total_canceled_quantity_per_product.sort_values(by=['Quantity'], ascendin

# Visualization
plt.figure(figsize=(12, 6))
sns.barplot(x='Quantity', y='Product', data=total_canceled_quantity_per_product.head(10), palette='rocket')
```

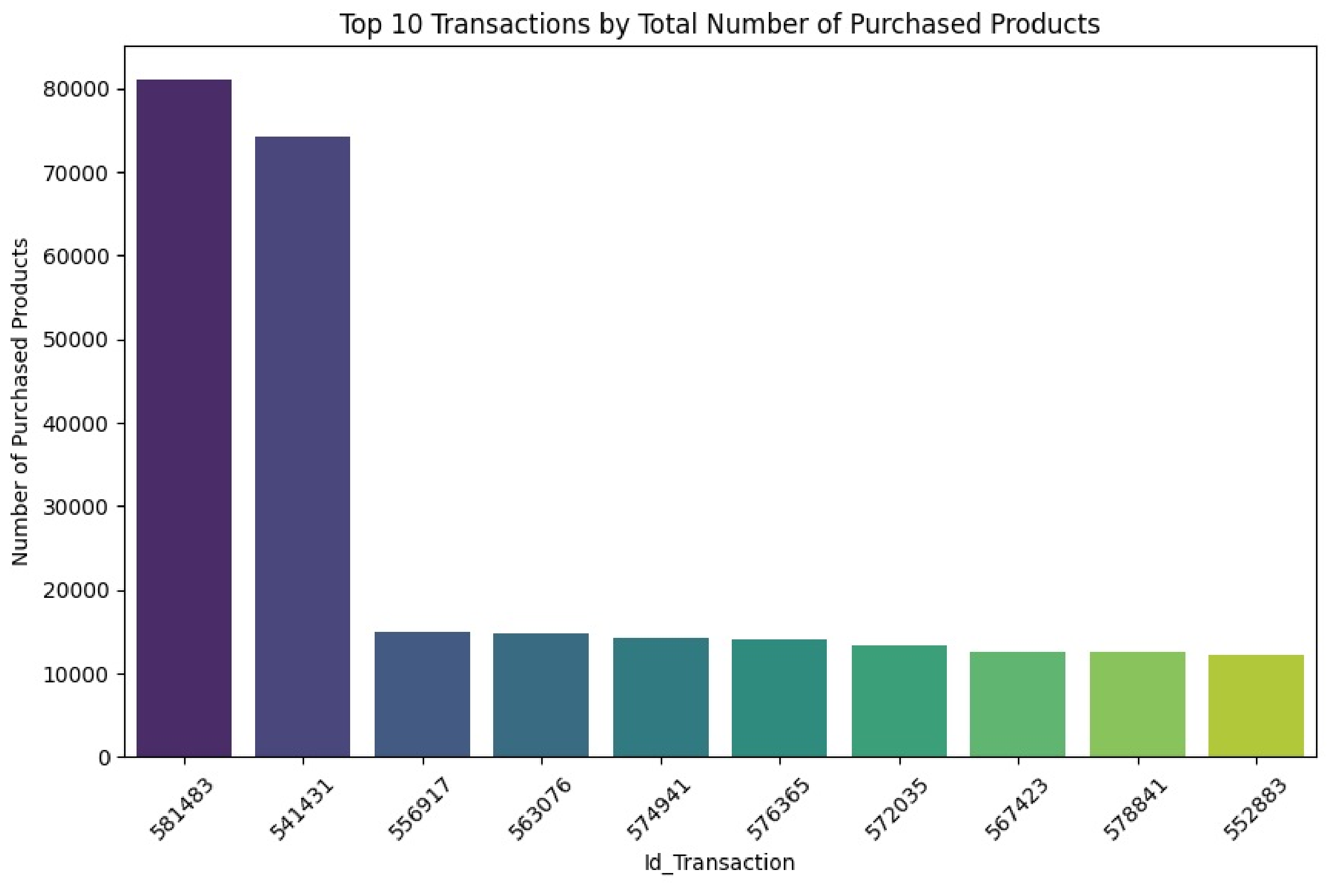
```
plt.xlabel('Total Canceled Quantity')
plt.ylabel('Product')
plt.title('Highest Canceled Products')
plt.show()
```



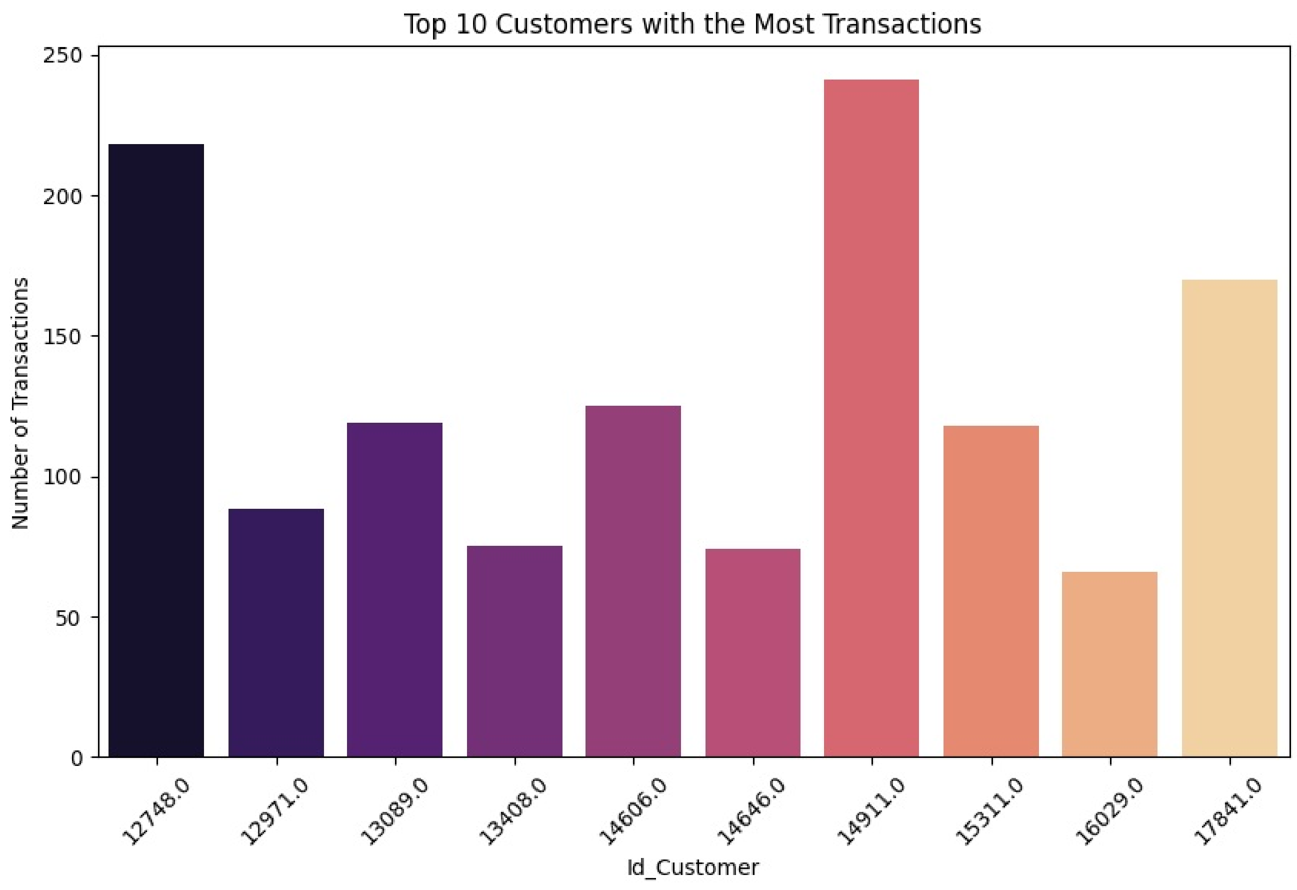
In [18]:
```
df_filtered = df[df['Quantity'] > 0]
average_quantity = round(df_filtered['Quantity'].mean())
print(f"Average Number of Products Purchased per Transaction : {average_quantity: }")
```

```
Average Number of Products Purchased per Transaction :  11
```

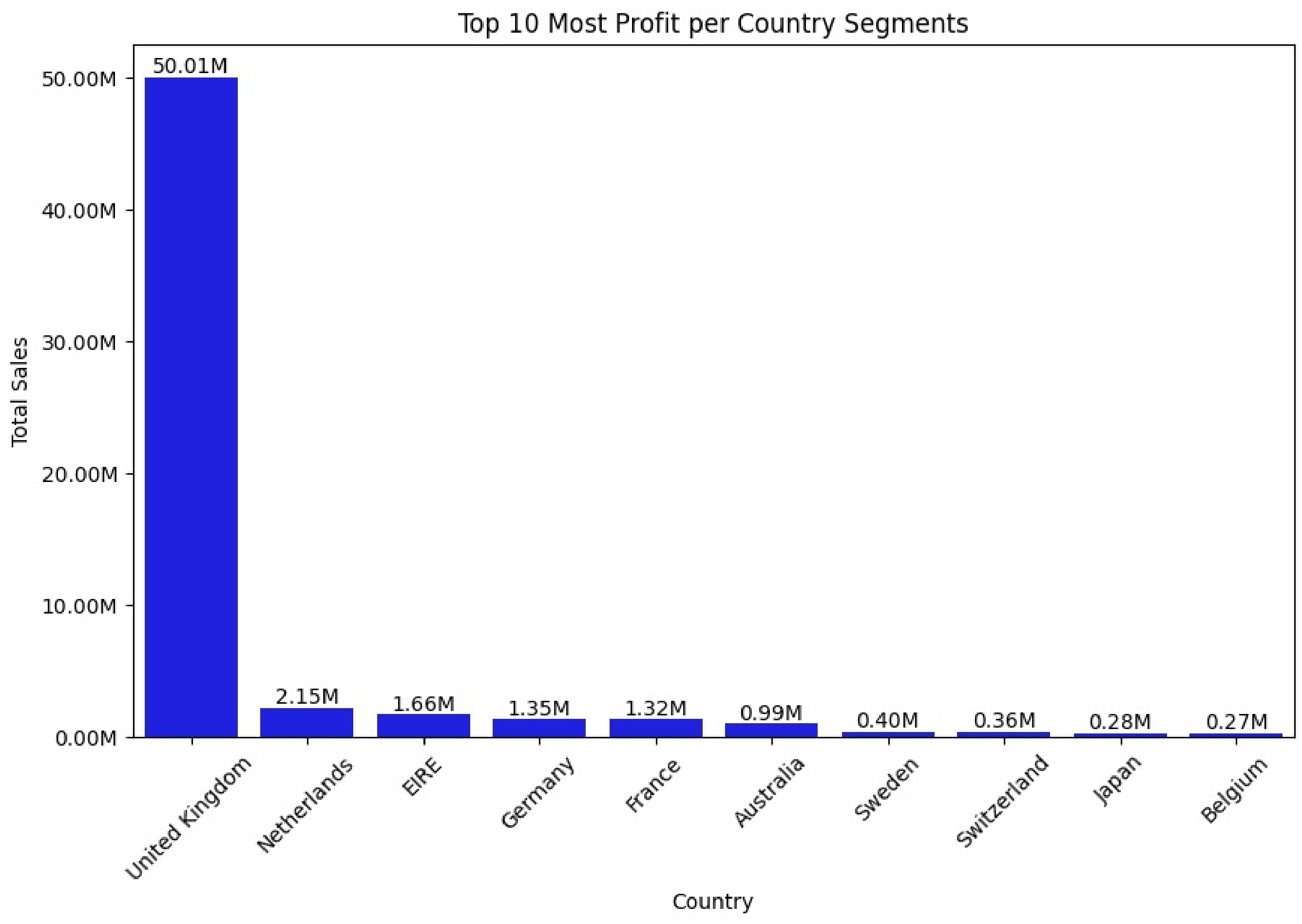
In [19]:
```
# To obtain the transaction IDs with the highest total purchased products
# it can group the data by transaction IDs and sum the Quantity for each transaction
total_purchased_per_transaction = df.groupby('Id_Transaction')['Quantity'].sum().reset_index()

# and then sort them in descending order for top 10 transactions by total number of purchased products
top_10 = total_purchased_per_transaction.sort_values(by='Quantity', ascending=False).head(10)

# Visualization
plt.figure(figsize=(10, 6))
sns.barplot(data=top_10, x='Id_Transaction', y='Quantity', palette='viridis')
plt.xlabel('Id_Transaction')
plt.ylabel('Number of Purchased Products')
plt.title('Top 10 Transactions by Total Number of Purchased Products')
plt.xticks(rotation=45)
plt.show()
```



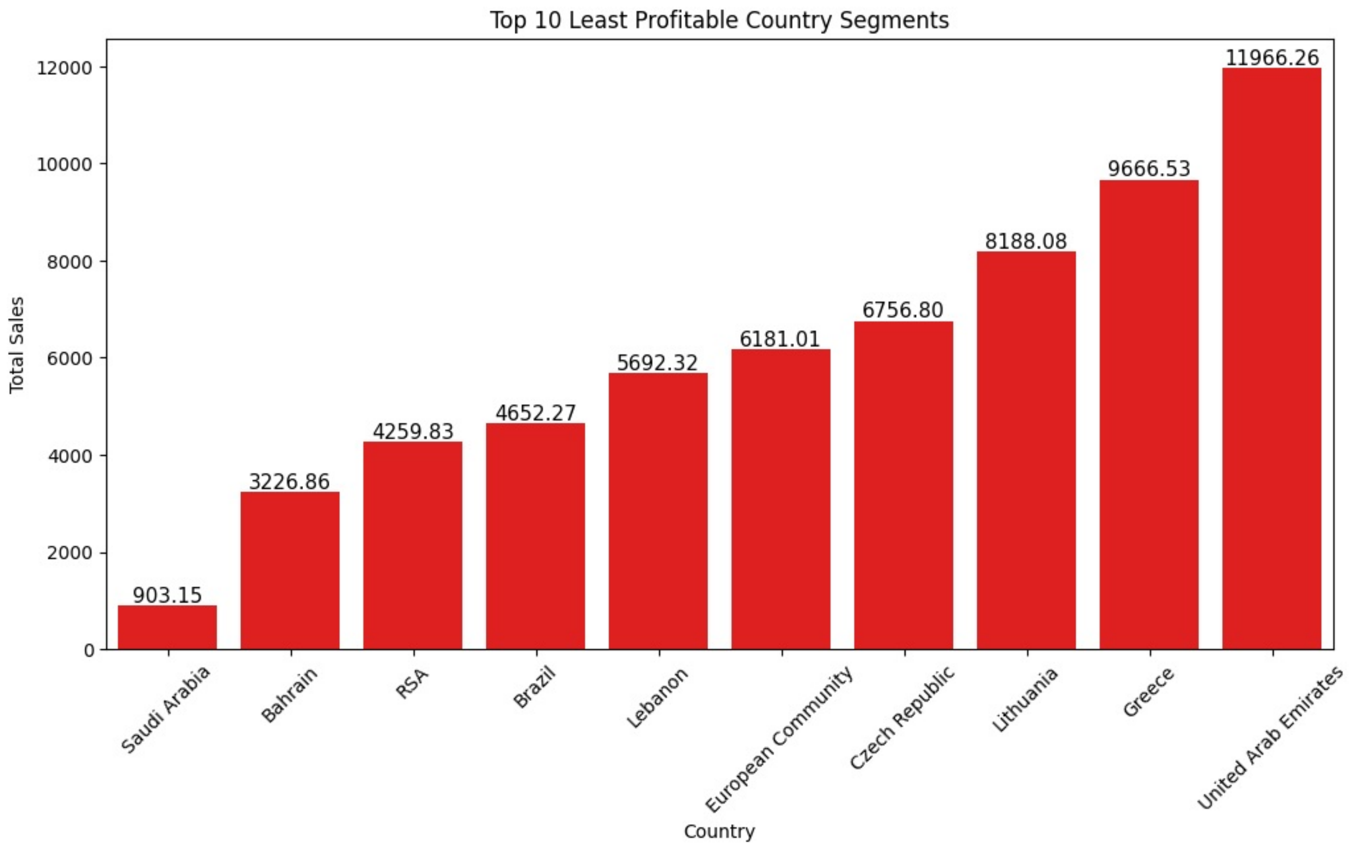
In [20]:
```
# To obtain the customers with the highest number of transactions
# it can group the data by customer IDs and count the unique transaction IDs for each customer
top_10_customers = df.groupby('Id_Customer')['Id_Transaction'].nunique().reset_index()

# and then sort them in descending order for top 10 customers with the most transactions
top_10_customers = top_10_customers.sort_values(by='Id_Transaction', ascending=False).head(10)

# Visualization
plt.figure(figsize=(10, 6))
sns.barplot(x='Id_Customer', y='Id_Transaction', data=top_10_customers, palette='magma')
plt.title('Top 10 Customers with the Most Transactions')
plt.xlabel('Id_Customer')
plt.ylabel('Number of Transactions')
plt.xticks(rotation=45)
plt.show()
```



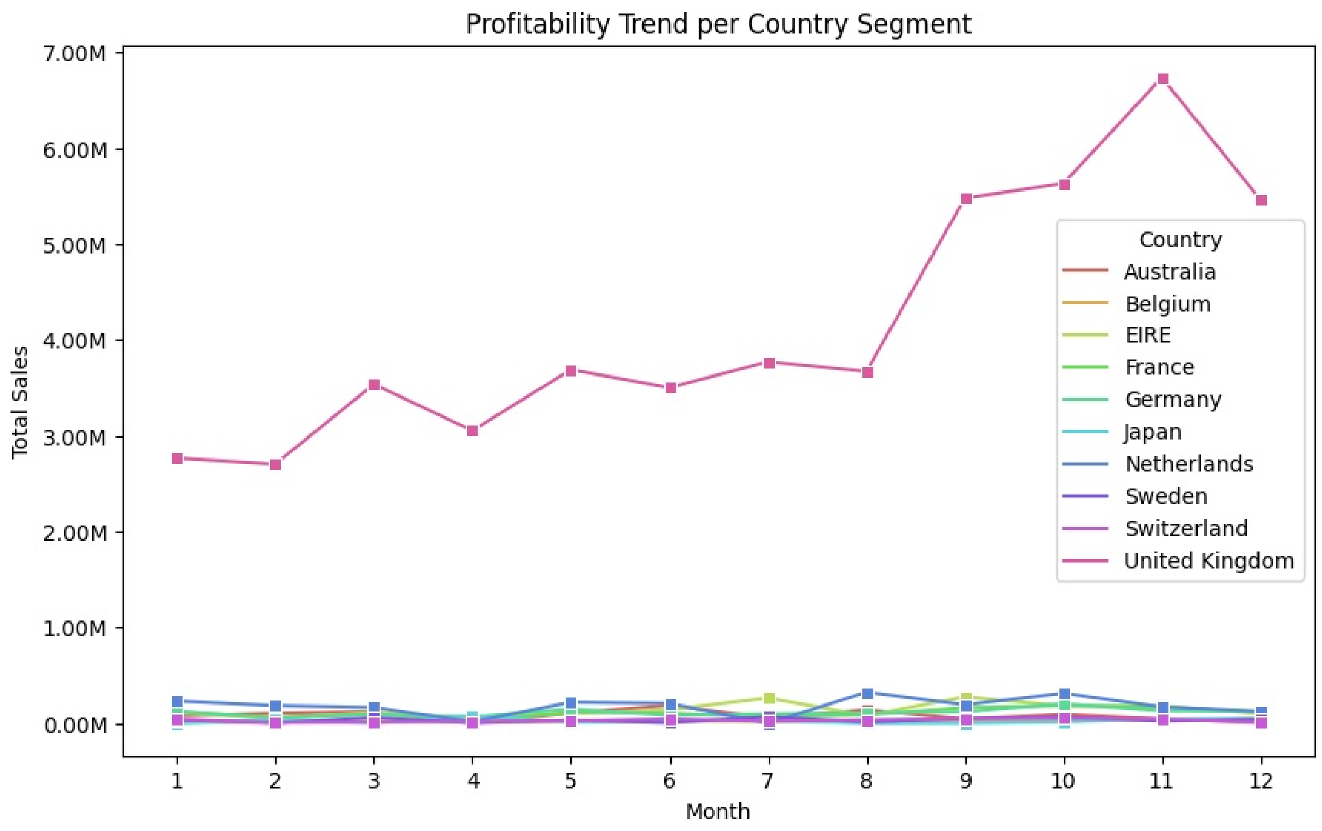
In [21]:
```
profit_per_segment = df.groupby('Country')['Total_Sales'].sum().reset_index()

# and then sort them in descending order for top 10 most profit per country segments
profit_per_segment = profit_per_segment.sort_values(by='Total_Sales', ascending=False).head(10)

def format_millions(value, _):
    return f'{value/1e6:.2f}M'

# Visualization
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='Country', y='Total_Sales', data=profit_per_segment, color="blue")
ax.yaxis.set_major_formatter(FuncFormatter(format_millions))
for p in ax.patches:
    height = p.get_height()
    ax.annotate(format_millions(height, None), (p.get_x() + p.get_width() / 2., height), ha='center', va='cente
plt.xlabel('Country')
plt.ylabel('Total Sales')
plt.title('Top 10 Most Profit per Country Segments')
plt.xticks(rotation=45)
plt.show()
```



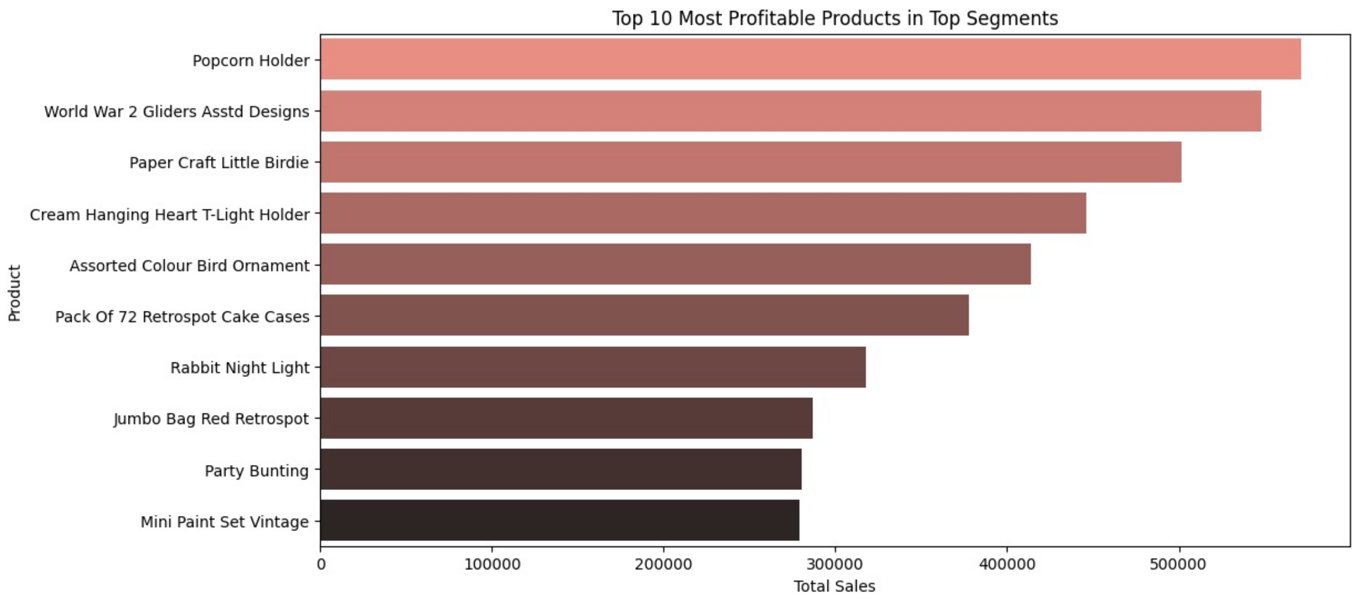
In [22]:
```
least_profitable = df.groupby('Country')['Total_Sales'].sum().reset_index()
least_profitable = least_profitable.sort_values(by='Total_Sales')
least_profitable = least_profitable.head(10)

# Visualization
plt.figure(figsize=(12, 6))
ax = sns.barplot(x='Country', y='Total_Sales', data=least_profitable, color='red')
plt.xlabel('Country')
plt.ylabel('Total Sales')
plt.title('Top 10 Least Profitable Country Segments')
plt.xticks(rotation=45)
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2f}', (p.get_x() + p.get_width() / 2., height), ha='center', va='center', fontsize=1
plt.show()
```



In [23]:
```
profit_per_month = df.groupby(['Country', df['Month']])['Total_Sales'].sum().reset_index()
most_profitable_segments = df.groupby('Country')['Total_Sales'].sum().reset_index()
most_profitable_segments = most_profitable_segments.sort_values(by='Total_Sales', ascending=False).head(10)
most_profitable_segments = most_profitable_segments['Country'].tolist()

# Filter the data to include only the most profitable segments
top_most_profitable_segments = profit_per_month[profit_per_month['Country'].isin(most_profitable_segments)]

# Visualization
plt.figure(figsize=(10, 6))
ax = sns.lineplot(x='Month', y='Total_Sales', hue='Country', data=top_most_profitable_segments, palette='hls', 
ax.yaxis.set_major_formatter(FuncFormatter(format_millions_y))
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.title('Profitability Trend per Country Segment')
plt.xticks(range(1, 13))
plt.show()
```



In [24]:
```python
# Calculate total sales per product for the most profitable customer segments
# Filter the data to include only the segments identified as the most profitable
profit_per_product = df[df['Country'].isin(most_profitable_segments)]['Total_Sales'].groupby(df['Product']).sum

# Sort the result by total sales in descending order
profit_per_product = profit_per_product.sort_values(by='Total_Sales', ascending=False).head(10)

plt.figure(figsize=(12, 6))
sns.barplot(x='Total_Sales', y='Product', data=profit_per_product, palette='dark:salmon_r')
plt.xlabel('Total Sales')
plt.ylabel('Product')
plt.title('Top 10 Most Profitable Products in Top Segments')
plt.show()
```



5. Based on your findings, what strategy could you recommend to the business to gain more profit?

1. Focus on Top Products: Prioritize and promote high-demand products like the "Popcorn Holder" to boost sales.
2. Optimize Sales Timing: Concentrate marketing efforts during months of increasing sales and offer promotions during slower months to maintain customer interest.
3. Customer Engagement: Implement loyalty programs and personalized marketing to retain high-value customers.
4. Market Expansion: Explore new markets, especially in less profitable regions like Saudi Arabia, using tailored strategies.
5. Product Diversification: Introduce related products to encourage additional purchases.

6. Discounts and Promotions: Use discounts and promotions strategically to stimulate demand.
7. Cost Control: Streamline operations and negotiate supplier deals to reduce costs.
8. Data-Driven Decisions: Utilize data analytics for insights into trends and customer preferences.

Also To address the issue of high cancellations of transactions and products, it is recommended to investigate the reasons behind these cancellations and implement measures to reduce them, potentially including improving product descriptions or packaging, and providing better customer support. These strategies can enhance profitability by optimizing sales, customer engagement, and cost management. Regular evaluation and adjustments are crucial for long-term success.