Cart abandonment occurs when shoppers add items to an online shopping cart but leave the site before completing the purchase. It's a critical issue for e-commerce businesses, with average abandonment rates around 70% (ranging from 60-80% depending on the industry)

Business Questions to Answer
Device-Specific Abandonment: What are the cart abandonment rates across different devices (desktop, mobile, tablet)?

Product-Specific Abandonment: Which product categories experience the highest rates of abandonment? Are there certain products that are frequently abandoned?

Time-Specific Trends: Are there specific times of the day, days of the week, or months where cart abandonment spikes? What seasonal trends exist?

In [7]:

```
# Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
import warnings
warnings.filterwarnings('ignore')
```

Import Datasets

In [8]:

```
customer=pd.read_csv('D:\\fullstackNaresh\\projects\\CartAbandonment\\customer_t
date=pd.read_csv('D:\\fullstackNaresh\\projects\\CartAbandonment\\date_table.csv
device=pd.read_csv('D:\\fullstackNaresh\\projects\\CartAbandonment\\device_table
product=pd.read_csv('D:\\fullstackNaresh\\projects\\CartAbandonment\\product_tab
fact=pd.read_csv('D:\\fullstackNaresh\\projects\\CartAbandonment\\fact_table.csv
```

Reading the datasets

In [9]:

```
customer.head()
```

Out[9]:

	customer_id	customer_name	age	gender	city
0	1	Customer 1	49	Female	London
1	2	Customer 2	45	Male	London
2	3	Customer 3	51	Male	London
3	4	Customer 4	38	Male	New York
4	5	Customer 5	26	Male	London

In [10]:

```
date.head()
```

Out[10]:

	date_id	date
0	1	1/1/2023
1	2	1/2/2023
2	3	1/3/2023
3	4	1/4/2023
4	5	1/5/2023

In [18]:

```
device.head()
```

Out[18]:

	device_id	device_type	os
0	1	Tablet	iOS
1	2	Desktop	iOS
2	3	Mobile	Windows
3	4	Mobile	Android
4	5	Tablet	iOS

In [17]:

```
product.head(0)
```

Out[17]:

product_id	product_name	category	price

In [16]:

```
fact.head(0)
```

Out[16]:

session_id	customer_id	product_id	device_id	date_id	quantity	abandonment_time

In [15]:

```
fact.head(0)
```

Out[15]:

session_id	customer_id	product_id	device_id	date_id	quantity	abandonment_time

Table Info and Data Cleaning

Data cleanup of date table is started.

In [19]:
```
date.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   date_id  366 non-null    int64 
 1   date     366 non-null    object
dtypes: int64(1), object(1)
memory usage: 5.8+ KB
```

Convert the object to to_datetime

In [20]:
```
date['date']=pd.to_datetime(date['date'])
date.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype         
---  ------   --------------  -----         
 0   date_id  366 non-null    int64         
 1   date     366 non-null    datetime64[ns]
dtypes: datetime64[ns](1), int64(1)
memory usage: 5.8 KB
```

```
=================================================
```

In [21]:
```
date=date[date['date'].dt.year==2023]
```

Data cleanup of fact table is started.

In [25]:
```
fact.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   session_id         5000 non-null   int64 
 1   customer_id        5000 non-null   int64 
 2   product_id         5000 non-null   int64 
 3   device_id          5000 non-null   int64 
 4   date_id            5000 non-null   int64 
 5   quantity           5000 non-null   int64 
 6   abandonment_time   2524 non-null   object
dtypes: int64(6), object(1)
memory usage: 273.6+ KB
```

6th Row is object , convert it into datetime64

In [26]:
```
fact['abandonment_time']=pd.to_datetime(fact['abandonment_time'])
fact.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   session_id        5000 non-null   int64         
 1   customer_id       5000 non-null   int64         
 2   product_id        5000 non-null   int64         
 3   device_id         5000 non-null   int64         
 4   date_id           5000 non-null   int64         
 5   quantity          5000 non-null   int64         
 6   abandonment_time  2524 non-null   datetime64[ns]
dtypes: datetime64[ns](1), int64(6)
memory usage: 273.6 KB
```

abandonment_time was object type and it is converted into datetime format

In [27]:
```
fact.isnull().sum()
```

Out[27]:
```
session_id             0
customer_id            0
product_id             0
device_id              0
date_id                0
quantity               0
abandonment_time    2476
dtype: int64
```

In [28]:
```
fact.duplicated().sum()
```

Out[28]:
```
0
```

Data Analysis

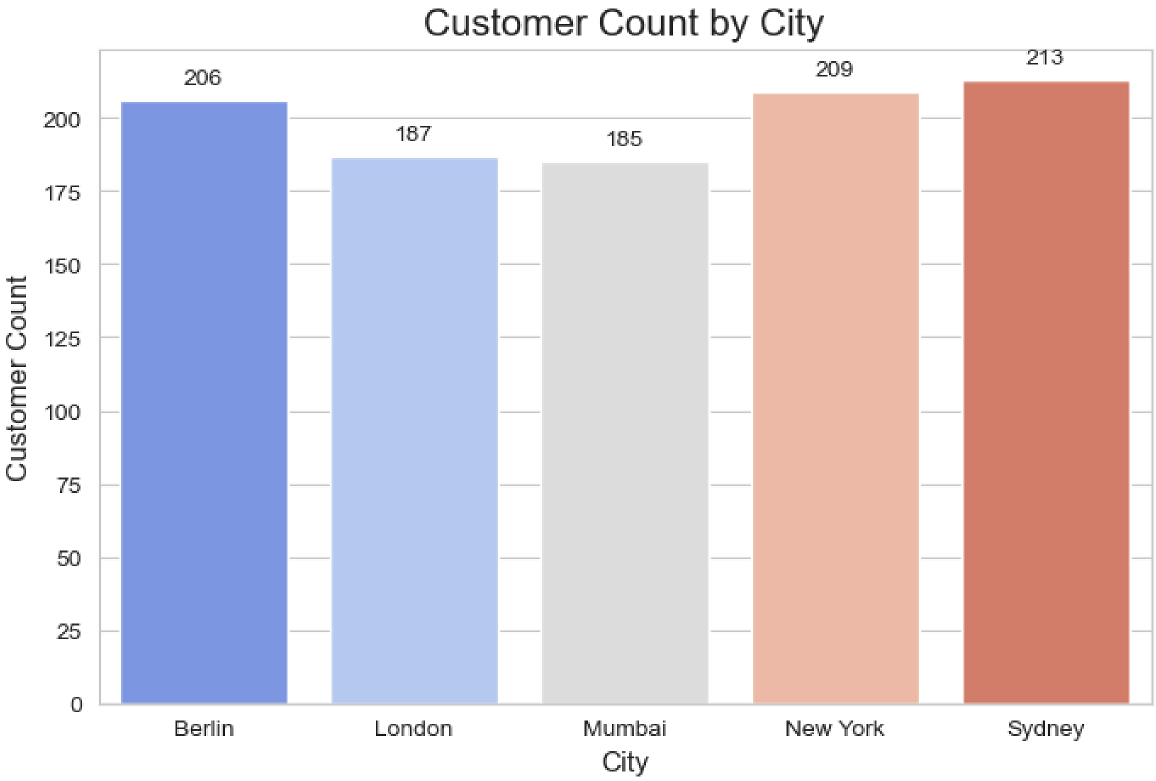
Demographic Insights

In [29]:
```
# Total Number of Customers
customer['customer_id'].count()
```

Out[29]:
```
1000
```

In [31]:
```
# City wise Customer Count
city_customer=customer.groupby(['city']).agg({'customer_id':'count'}).rename(col

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Create the bar plot
plt.figure(figsize=(8,5))
barplot = sns.barplot(x='city', y='customer count', data=city_customer, palette=

# Add data labels on the bars
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  # Format the label
                     (p.get_x() + p.get_width() / 2., p.get_height()),  # Positi
```

```
                      ha='center', va='center', xytext=(0, 9), textcoords='offset

# Add title and labels
plt.title('Customer Count by City', fontsize=16)
plt.xlabel('City', fontsize=12)
plt.ylabel('Customer Count', fontsize=12)

# Display plot
plt.show()
```

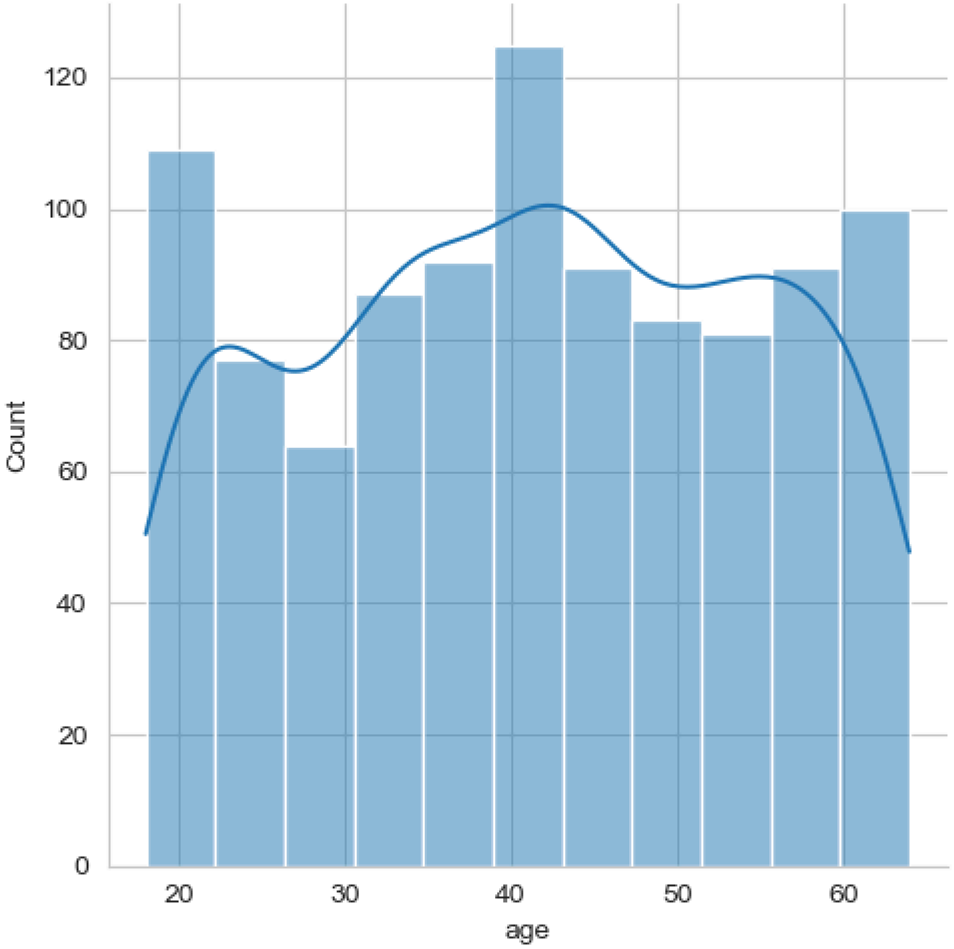


Sydney has highest number of customers(213) New York (209),Berlin(206), London(187) and Mumbai(185)

So Sydney has highest number of customers and Mumbai has lowest number of customers.

In [32]:
```
# Age Distribution of All the Customers
sns.displot(customer['age'],kde=True)
```

Out[32]:
```
<seaborn.axisgrid.FacetGrid at 0x18dd6969910>
```



From the above age distribution chart we can see most of customers age is 40

In [33]:

```
# Genderwise Customer Distribution
gender_df=customer.groupby(['gender']).agg({'customer_id':'count'}).rename(colum
gender_df
```

Out[33]:

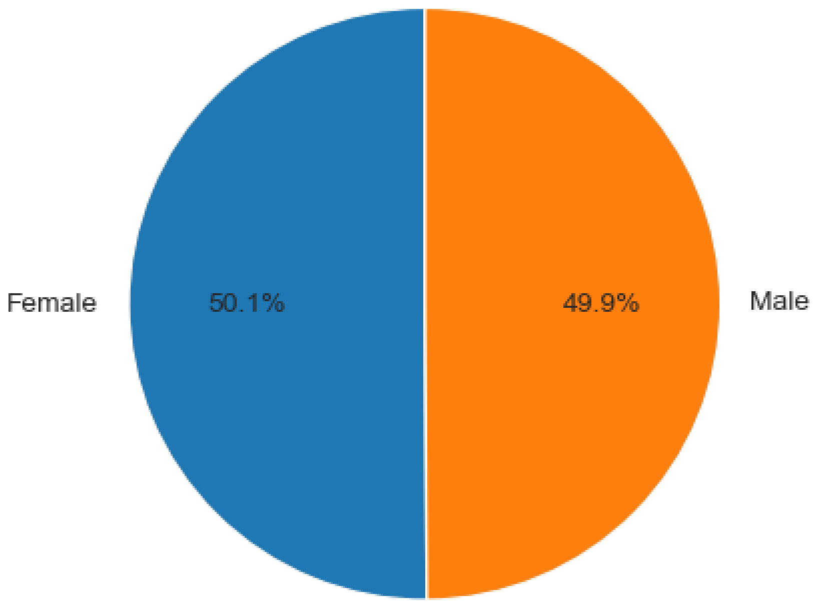
	gender	customer count
0	Female	501
1	Male	499

In [34]:

```
plt.pie(gender_df['customer count'], labels=gender_df['gender'], autopct='%1.1f%
```

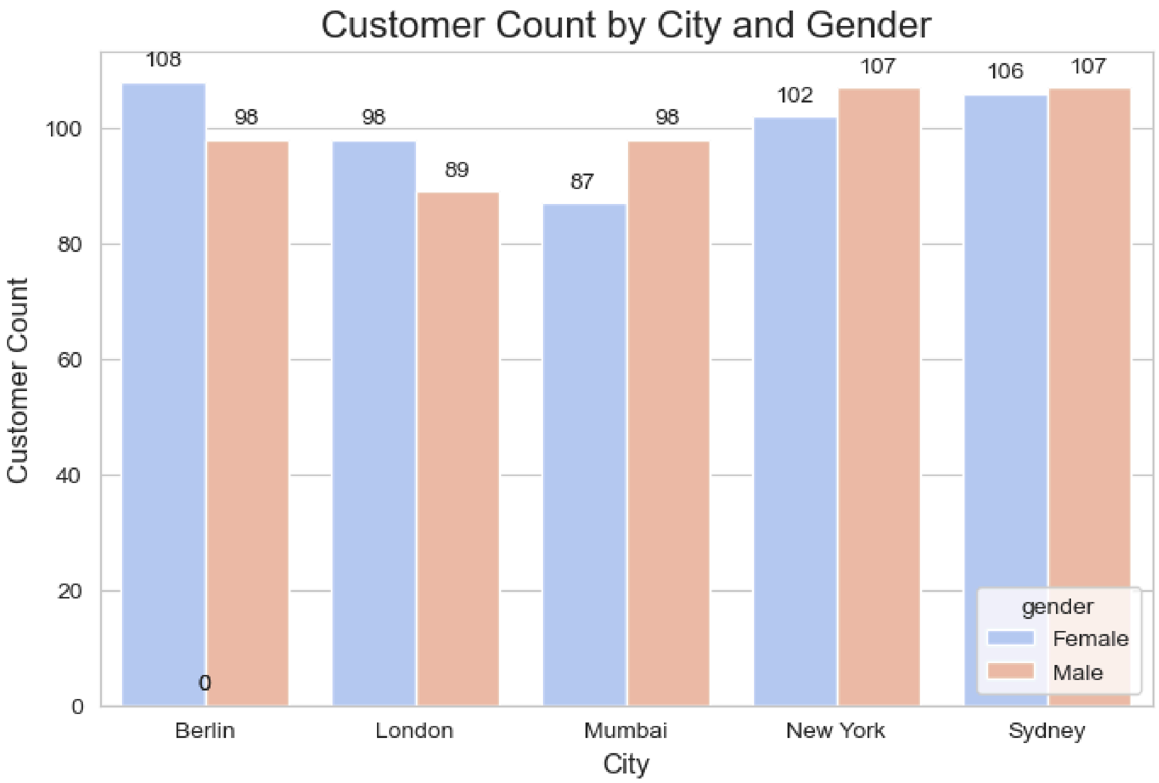
Out[34]:

```
([<matplotlib.patches.Wedge at 0x18dd5d4fb00>,
  <matplotlib.patches.Wedge at 0x18dd5d4c8c0>],
 [Text(-1.099994571861817, -0.0034557017432520097, 'Female'),
  Text(1.099994571861817, 0.0034557017432518744, 'Male')],
 [Text(-0.5999970391973546, -0.0018849282235920048, '50.1%'),
  Text(0.5999970391973546, 0.0018849282235919313, '49.9%')])
```



In [35]:
```
city_gender=customer.groupby(['city','gender']).agg({'customer_id':'count'}).ren
```

In [36]:
```
# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Create the bar plot
plt.figure(figsize=(8, 5))
barplot = sns.barplot(x='city', y='customer count', hue='gender', data=city_gend

# Add data labels on the bars
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  # Format the label
                     (p.get_x() + p.get_width() / 2., p.get_height()),  # Positi
                     ha='center', va='center', xytext=(0, 9), textcoords='offset

# Add title and labels
plt.title('Customer Count by City and Gender', fontsize=16)
plt.xlabel('City', fontsize=12)
plt.ylabel('Customer Count', fontsize=12)

# Display plot
plt.show()
```



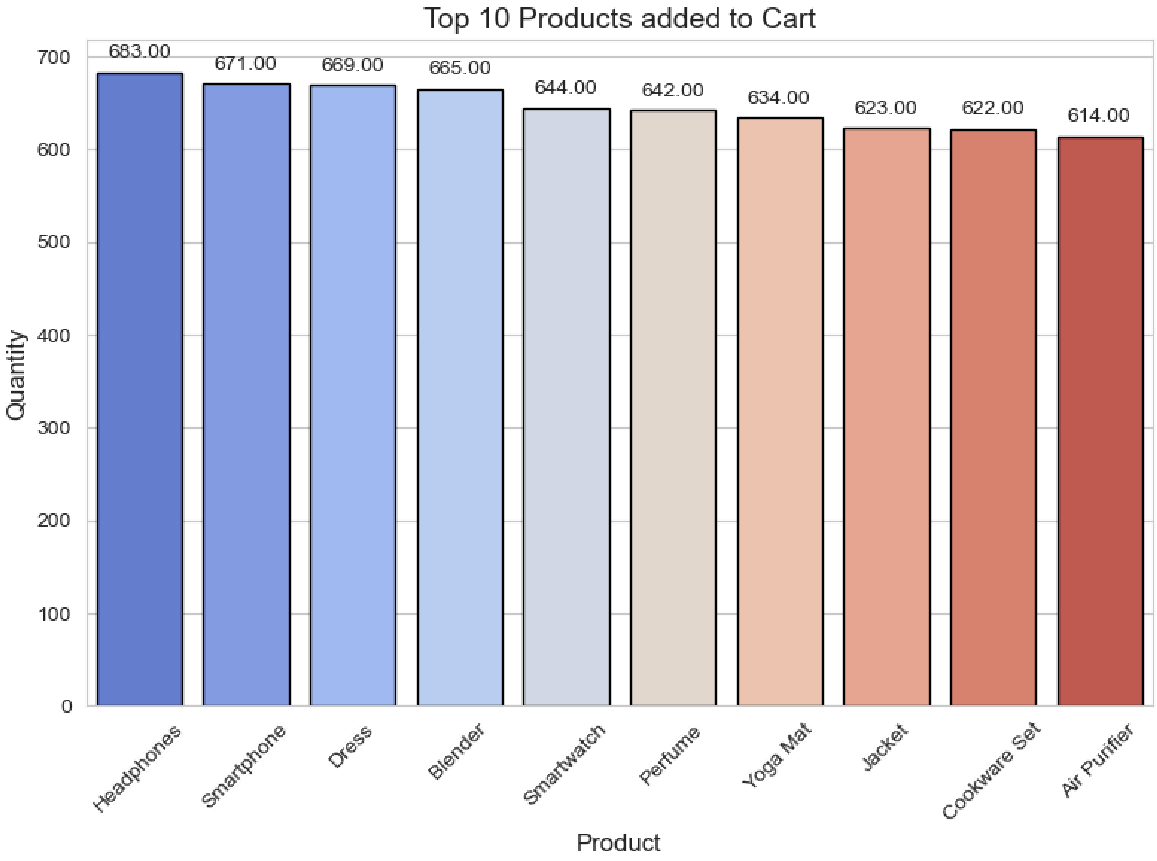
Berlin & London has more Female customers Mumbai, New york has more Male customers Sydney has almost same male and female customers

Cart Analysis

In [38]:

```
product_cart=pd.merge(product,fact,how='inner',on='product_id')
product_cart_qty=product_cart.groupby(['product_name']).agg({'quantity':'sum'}).
product_cart_qty['Rank']=product_cart_qty['quantity'].rank(method='dense',ascend
top_10_product=product_cart_qty[product_cart_qty['Rank']<=10].sort_values(by=['R

# Setting figure size
plt.figure(figsize=(8,6))

# Creating bar plot using seaborn
ax = sns.barplot(x='product_name', y='quantity', data=top_10_product, palette='c

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Adding title and labels
plt.title('Top 10 Products added to Cart', fontsize=14)
plt.xlabel('Product', fontsize=12)
plt.ylabel('Quantity', fontsize=12)

# Adding values on top of bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9), textcoords = 'offset points')
# Display the plot
```

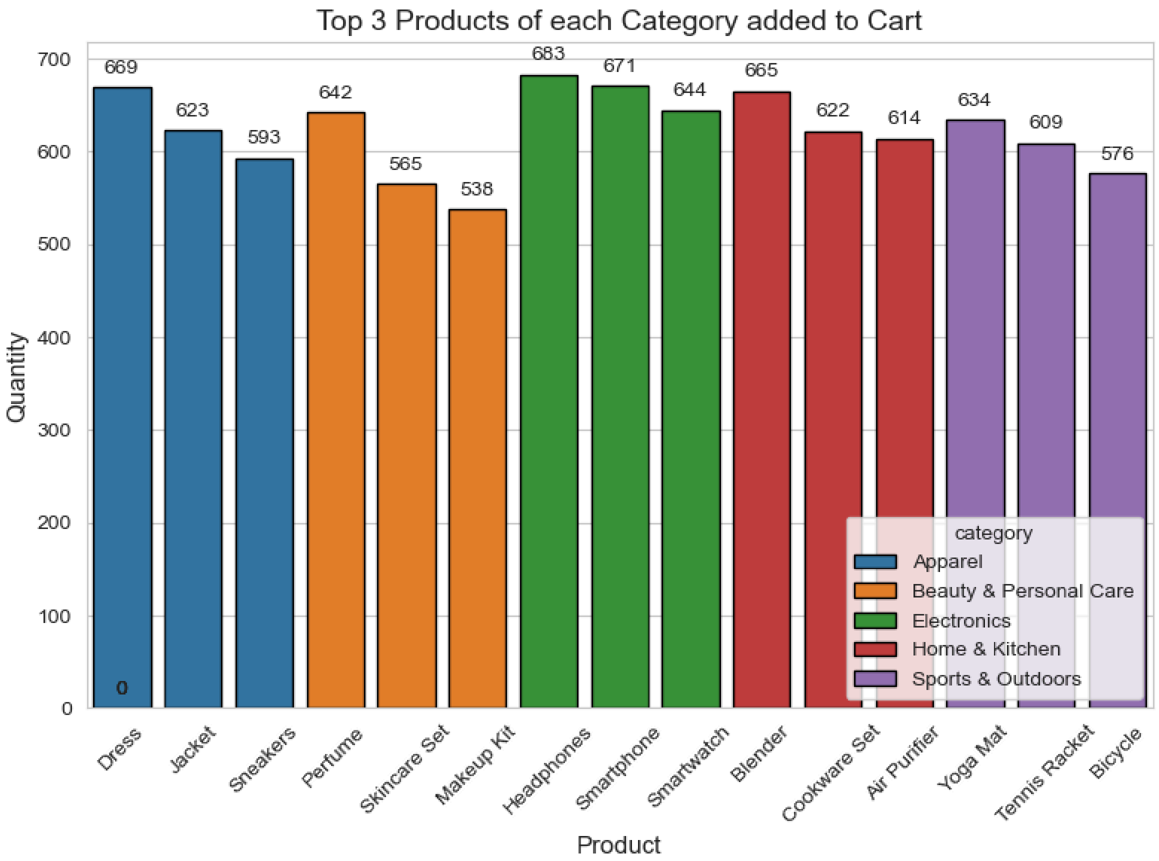
```
plt.tight_layout()
plt.show()
```



Above the bar chart shows top 10 products added to cart. From we can see

Headphones,Smartphone is top 2 product that added to cart and these two belongs from same category i.e Electronics. Smartwatch is also in top 10 products added to cart. If we can combinatio of these theree products it could increase sales.

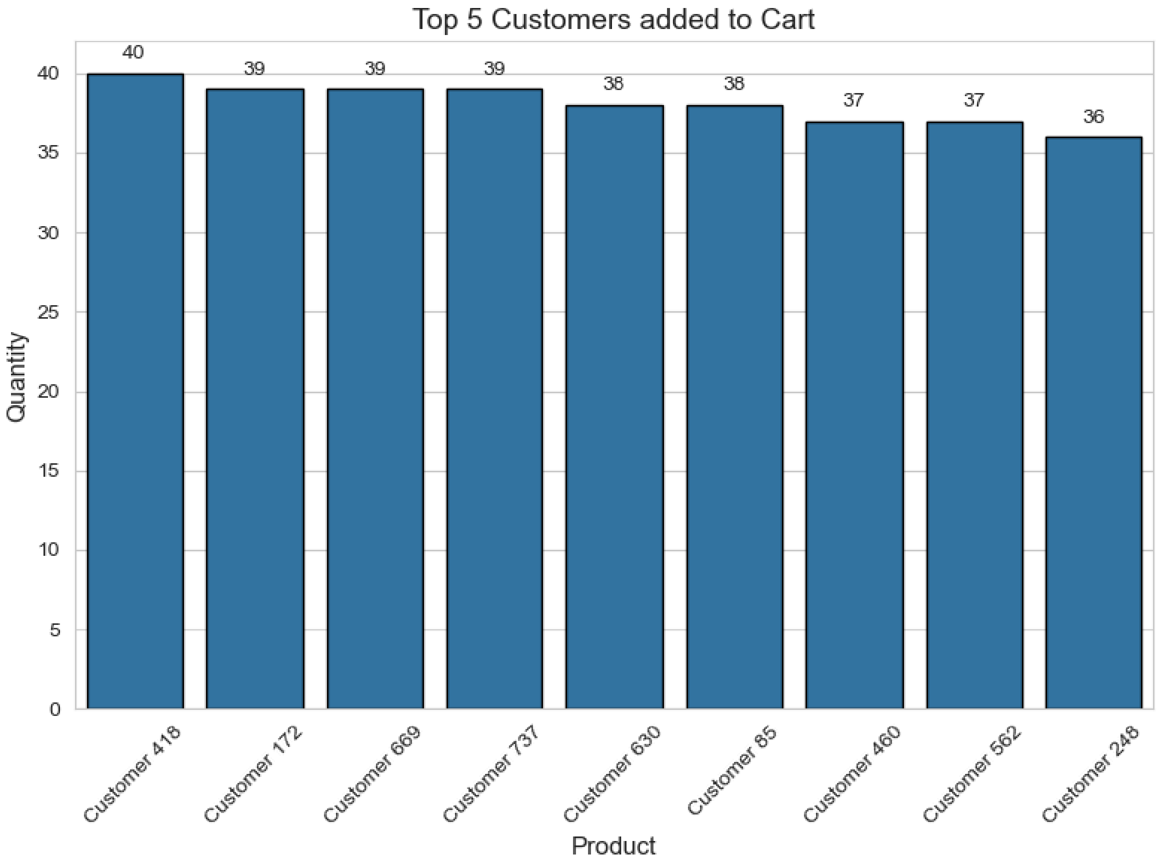
In [39]:
```
# Top 3 Products of each category added to cart
category_product_cart=product_cart.groupby(['category','product_name']).agg({'qu
category_product_cart['Rank']=category_product_cart.groupby(['category'])['quant
top_2_cateory_product=category_product_cart[category_product_cart['Rank']<=3].sc

# Setting figure size
plt.figure(figsize=(8,6))

# Creating bar plot using seaborn
ax = sns.barplot(x='product_name', y='quantity', data=top_2_cateory_product, hue

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Adding title and labels
plt.title('Top 3 Products of each Category added to Cart', fontsize=14)
plt.xlabel('Product', fontsize=12)
plt.ylabel('Quantity', fontsize=12)

# Adding values on top of bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9), textcoords = 'offset points')

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



Electronics and Home & Kitchen categories are seeing strong demand, with products like Headphones, Smartphones, and Blenders leading. Apparel and Beauty & Personal Care categories also have consistent sales across several products, particularly Dresses, Perfume, and Makeup Kits. Sports & Outdoors products show increasing interest, particularly with the Yoga Mat being the top product, reflecting a growing trend in fitness and wellness.

In [41]:

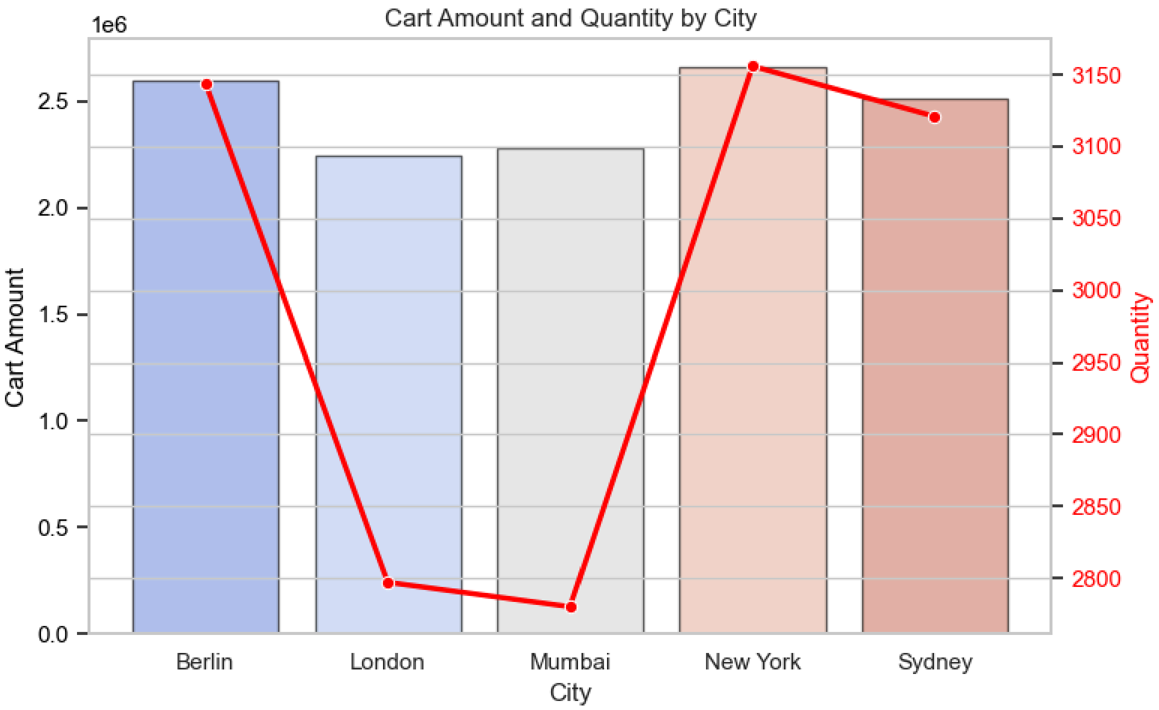
```
customer_cart=pd.merge(customer,fact,how='inner',on='customer_id')
customer_cart_qty=customer_cart.groupby(['customer_name']).agg({'quantity':'sum'
customer_cart_qty['Rank']=customer_cart_qty['quantity'].rank(method='dense',asce
top_customer_cart=customer_cart_qty[customer_cart_qty['Rank']<=5].sort_values(by

# Setting figure size
plt.figure(figsize=(8,6))

# Creating bar plot using seaborn
ax = sns.barplot(x='customer_name', y='quantity', data=top_customer_cart, edgeco

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Adding title and labels
plt.title('Top 5 Customers added to Cart', fontsize=14)
plt.xlabel('Product', fontsize=12)
plt.ylabel('Quantity', fontsize=12)

# Adding values on top of bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9), textcoords = 'offset points')

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



In [43]:

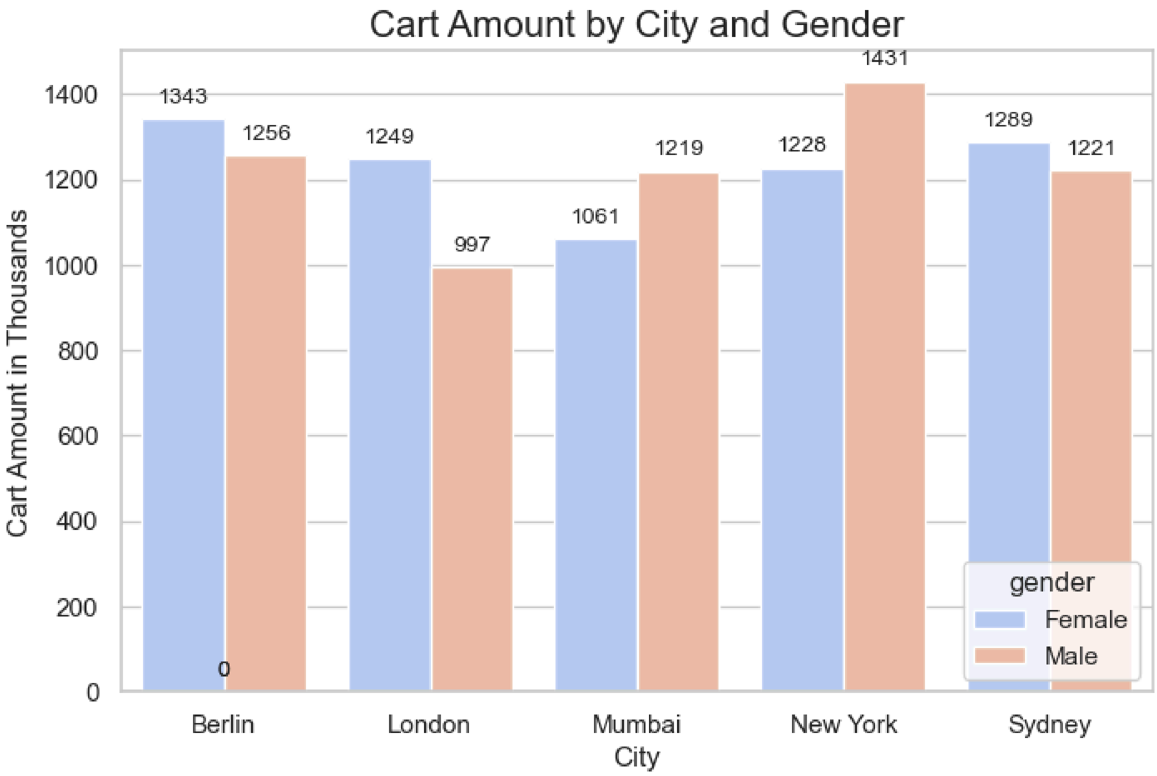
```
product_cart['cart amount'] = product_cart['quantity'] * product_cart['price']
customer_product_cart=pd.merge(product_cart,customer,how='inner',on='customer_id
city_cart=customer_product_cart.groupby(['city']).agg({'quantity':'sum','cart am

# Set the style of seaborn
sns.set(style="whitegrid")

# Create a figure and a set of subplots
fig, ax1 = plt.subplots(figsize=(8, 5))

# Create bar plot for cart amount
sns.barplot(x='city', y='cart amount', data=city_cart, palette='coolwarm', edgec
ax1.set_ylabel('Cart Amount', color='black')
ax1.tick_params(axis='y', labelcolor='black')
ax1.grid(False)

# Create a second y-axis for the quantity line plot
ax2 = ax1.twinx()
sns.lineplot(x='city', y='quantity', data=city_cart, color='red', marker='o', ax
ax2.set_ylabel('Quantity', color='red')
```

```
ax2.tick_params(axis='y', labelcolor='red')

# Add titles and labels
plt.title('Cart Amount and Quantity by City')
ax1.set_xlabel('City')

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



In [44]:

```
city_gender_cartamount=customer_product_cart.groupby(['city','gender']).agg({'ca
city_gender_cartamount['cart amount thousands']=city_gender_cartamount['cart amo
city_gender_cartamount['cart amount thousands']=city_gender_cartamount['cart amo

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Create the bar plot
plt.figure(figsize=(8, 5))
barplot = sns.barplot(x='city', y='cart amount thousands', hue='gender', data=ci

# Add data labels on the bars
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  # Format the label
                     (p.get_x() + p.get_width() / 2., p.get_height()),  # Positi
                     ha='center', va='center', xytext=(0, 9), textcoords='offset

# Add title and labels
plt.title('Cart Amount by City and Gender', fontsize=16)
plt.xlabel('City', fontsize=12)
plt.ylabel('Cart Amount in Thousands', fontsize=12)

# Display plot
plt.show()
```

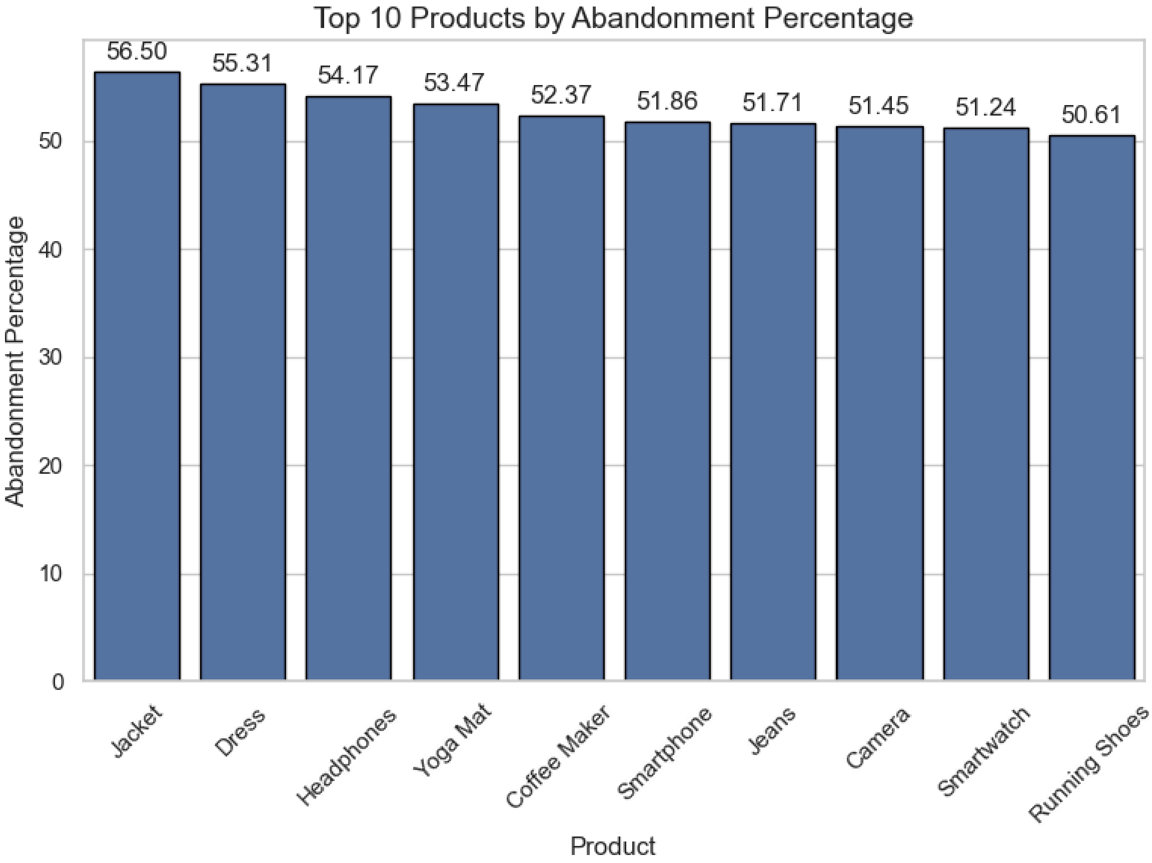


For Berlin,London and Sydney Cart quantity is higer for female. For Mumbai,New York Cart Quantity is higer for male.

In [45]:
```
customer_product_cart['abandonment qty']=np.where(pd.isna(customer_product_cart[
```

Cart Abandonment Rate

In [46]:
```
# Product-Specific Cart Abandonment Rate
prodcut_abandonment=customer_product_cart.groupby(['product_name']).agg({'quanti
prodcut_abandonment['abandonment rate']=prodcut_abandonment['abandonment qty']*1
prodcut_abandonment['abandonment rate']=prodcut_abandonment['abandonment rate'].
prodcut_abandonment['Rank']=prodcut_abandonment['abandonment rate'].rank(method=
top_10_product_abandonment_rate=prodcut_abandonment[prodcut_abandonment['Rank']<

# Setting figure size
plt.figure(figsize=(8,6))

# Creating bar plot using seaborn
ax = sns.barplot(x='product_name', y='abandonment rate', data=top_10_product_aba

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Adding title and labels
plt.title('Top 10 Products by Abandonment Percentage', fontsize=14)
plt.xlabel('Product', fontsize=12)
plt.ylabel('Abandonment Percentage', fontsize=12)

# Adding values on top of bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9), textcoords = 'offset points')

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

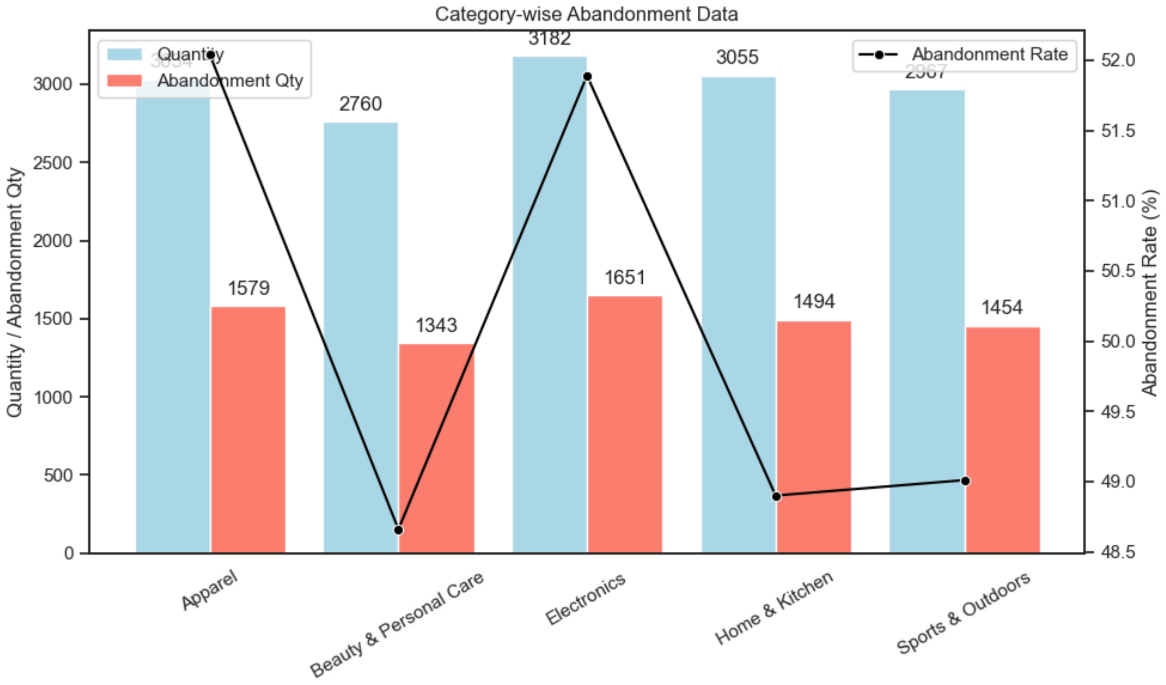


In [47]:

```
# Categorywise Abandnoment Rate
category_abandnoment=customer_product_cart.groupby(['category']).agg({'quantity'
category_abandnoment['abandonment rate']=category_abandnoment['abandonment qty']
category_abandnoment['abandonment rate']=category_abandnoment['abandonment rate'
```

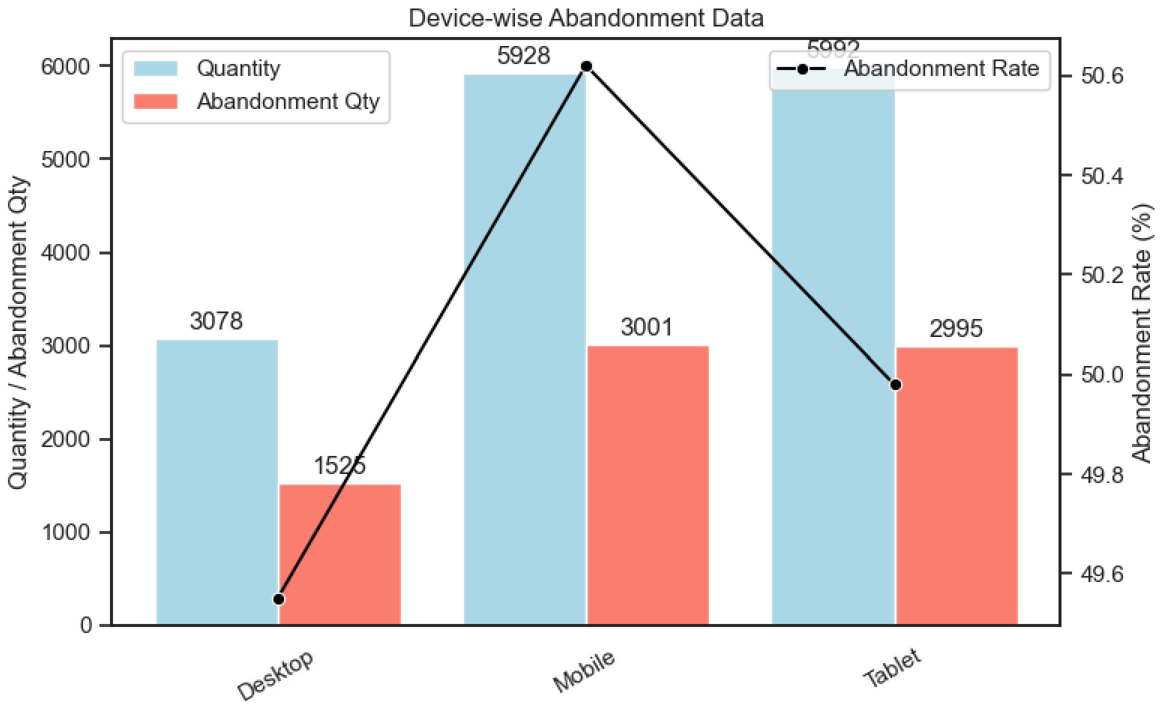
In [48]:

```
# Set the plot style without gridlines
sns.set(style="white")

# Create the figure and axis
fig, ax1 = plt.subplots(figsize=(10, 6))

# Set bar width and positions for side-by-side bars
bar_width = 0.4
index = np.arange(len(category_abandnoment['category']))

# Plot side-by-side bar charts for 'quantity' and 'abandonment_qty'
bars1 = ax1.bar(index - bar_width/2, category_abandnoment['quantity'], bar_width
bars2 = ax1.bar(index + bar_width/2, category_abandnoment['abandonment qty'], ba

# Set x-ticks and labels
ax1.set_xticks(index)
ax1.set_xticklabels(category_abandnoment['category'], rotation=30)

# Add data labels on top of the bars
for bar in bars1:
    yval = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, yval + 50, int(yval), ha='center',

for bar in bars2:
    yval = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, yval + 50, int(yval), ha='center',

# Create a second y-axis to plot the abandonment rate
ax2 = ax1.twinx()
sns.lineplot(x=index, y='abandonment rate', data=category_abandnoment, color='bl

# Remove gridlines from both axes
ax1.grid(False)
ax2.grid(False)

# Add labels and title
ax1.set_ylabel('Quantity / Abandonment Qty')
ax2.set_ylabel('Abandonment Rate (%)')
plt.title('Category-wise Abandonment Data')

# Add legends
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

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



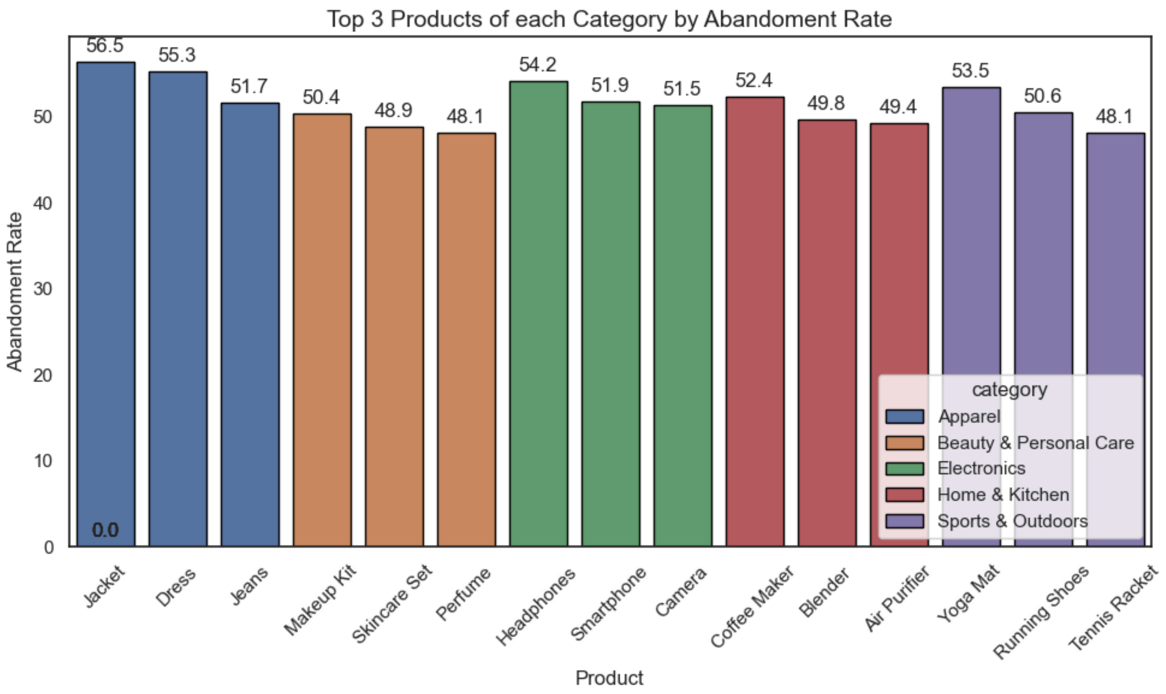
In [49]:
```
# Device-Specific Cart Abandonment Rate
customer_product_device_cart=pd.merge(device,customer_product_cart,how='inner',c
customer_product_device_abandnoment=customer_product_device_cart.groupby(['devic
customer_product_device_abandnoment['abandonment rate']=customer_product_device_
customer_product_device_abandnoment['abandonment rate']=customer_product_device_

# Set the plot style without gridlines
sns.set(style="white")

# Create the figure and axis
fig, ax1 = plt.subplots(figsize=(8, 5))
```

```
# Set bar width and positions for side-by-side bars
bar_width = 0.4
index = np.arange(len(customer_product_device_abandnoment['device_type']))

# Plot side-by-side bar charts for 'quantity' and 'abandonment_qty'
bars1 = ax1.bar(index - bar_width/2, customer_product_device_abandnoment['quanti
bars2 = ax1.bar(index + bar_width/2, customer_product_device_abandnoment['abandc

# Set x-ticks and labels
ax1.set_xticks(index)
ax1.set_xticklabels(customer_product_device_abandnoment['device_type'], rotation

# Add data labels on top of the bars
for bar in bars1:
    yval = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, yval + 50, int(yval), ha='center',

for bar in bars2:
    yval = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, yval + 50, int(yval), ha='center',

# Create a second y-axis to plot the abandonment rate
ax2 = ax1.twinx()
sns.lineplot(x=index, y='abandonment rate', data=customer_product_device_abandnc

# Remove gridlines from both axes
ax1.grid(False)
ax2.grid(False)

# Add labels and title
ax1.set_ylabel('Quantity / Abandonment Qty')
ax2.set_ylabel('Abandonment Rate (%)')
plt.title('Device-wise Abandonment Data')

# Add legends
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

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



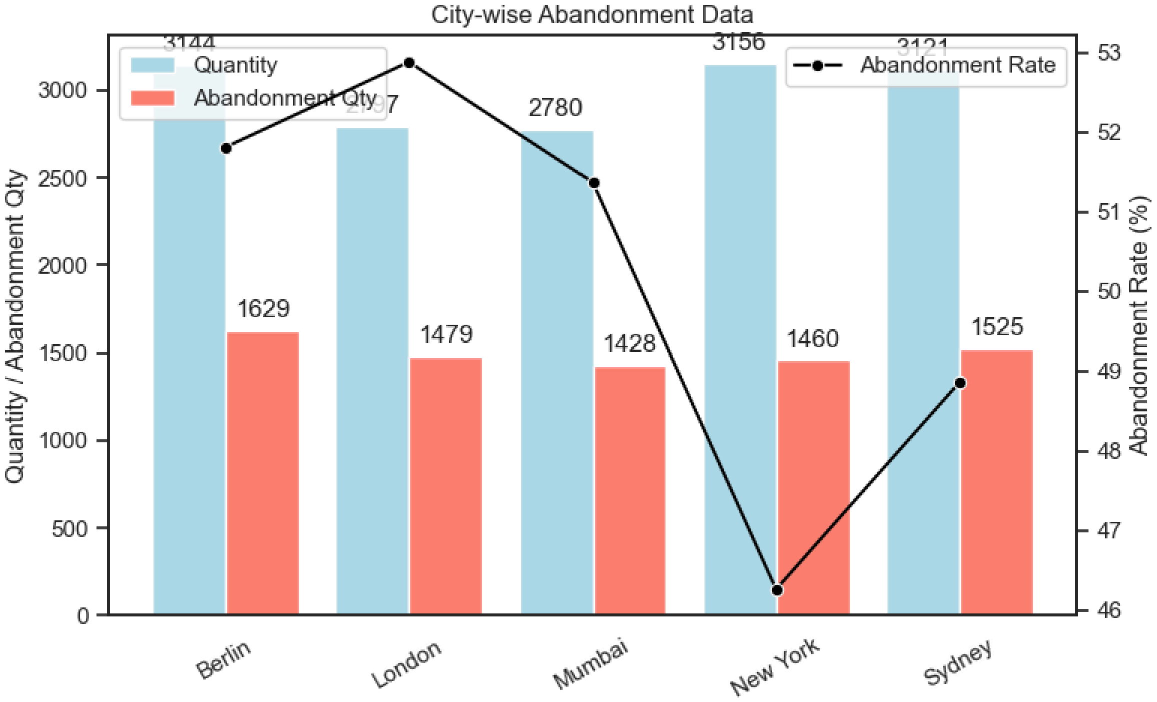
In [50]:

```python
# Top 3 Product of Each Category By Abandnoment Rate
category_product_abandnoment=customer_product_cart.groupby(['category','product_
category_product_abandnoment['abandonment rate']=category_product_abandnoment['a
category_product_abandnoment['abandonment rate']=category_product_abandnoment['a
category_product_abandnoment['Rank']=category_product_abandnoment.groupby(['cate
top_3_abandnoment_rate=category_product_abandnoment[category_product_abandnoment

# Setting figure size
plt.figure(figsize=(10,6))

# Creating bar plot using seaborn
ax = sns.barplot(x='product_name', y='abandonment rate', data=top_3_abandnoment_

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Adding title and labels
plt.title('Top 3 Products of each Category by Abandoment Rate', fontsize=14)
plt.xlabel('Product', fontsize=12)
plt.ylabel('Abandoment Rate', fontsize=12)

# Adding values on top of bars
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.1f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9), textcoords = 'offset points')

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



In [51]:
```
# Customer-Specific Cart Abandonment Rate
city_CAR=customer_product_cart.groupby(['city']).agg({'quantity':'sum','abandonm
city_CAR['abandonment rate']=city_CAR['abandonment qty']*100/city_CAR['quantity'
city_CAR['abandonment rate']=city_CAR['abandonment rate'].round(2)

# Set the plot style without gridlines
sns.set(style="white")

# Create the figure and axis
fig, ax1 = plt.subplots(figsize=(8, 5))

# Set bar width and positions for side-by-side bars
bar_width = 0.4
index = np.arange(len(city_CAR['city']))

# Plot side-by-side bar charts for 'quantity' and 'abandonment_qty'
bars1 = ax1.bar(index - bar_width/2, city_CAR['quantity'], bar_width, label='Qua
bars2 = ax1.bar(index + bar_width/2, city_CAR['abandonment qty'], bar_width, lab

# Set x-ticks and labels
ax1.set_xticks(index)
ax1.set_xticklabels(city_CAR['city'], rotation=30)

# Add data labels on top of the bars
for bar in bars1:
    yval = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, yval + 50, int(yval), ha='center',

for bar in bars2:
    yval = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, yval + 50, int(yval), ha='center',

# Create a second y-axis to plot the abandonment rate
ax2 = ax1.twinx()
sns.lineplot(x=index, y='abandonment rate', data=city_CAR, color='black', marker
# Remove gridlines from both axes
ax1.grid(False)
ax2.grid(False)
```

```
# Add labels and title
ax1.set_ylabel('Quantity / Abandonment Qty')
ax2.set_ylabel('Abandonment Rate (%)')
plt.title('City-wise Abandonment Data')

# Add legends
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

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



Potential Factors Influencing Abandonment by City:

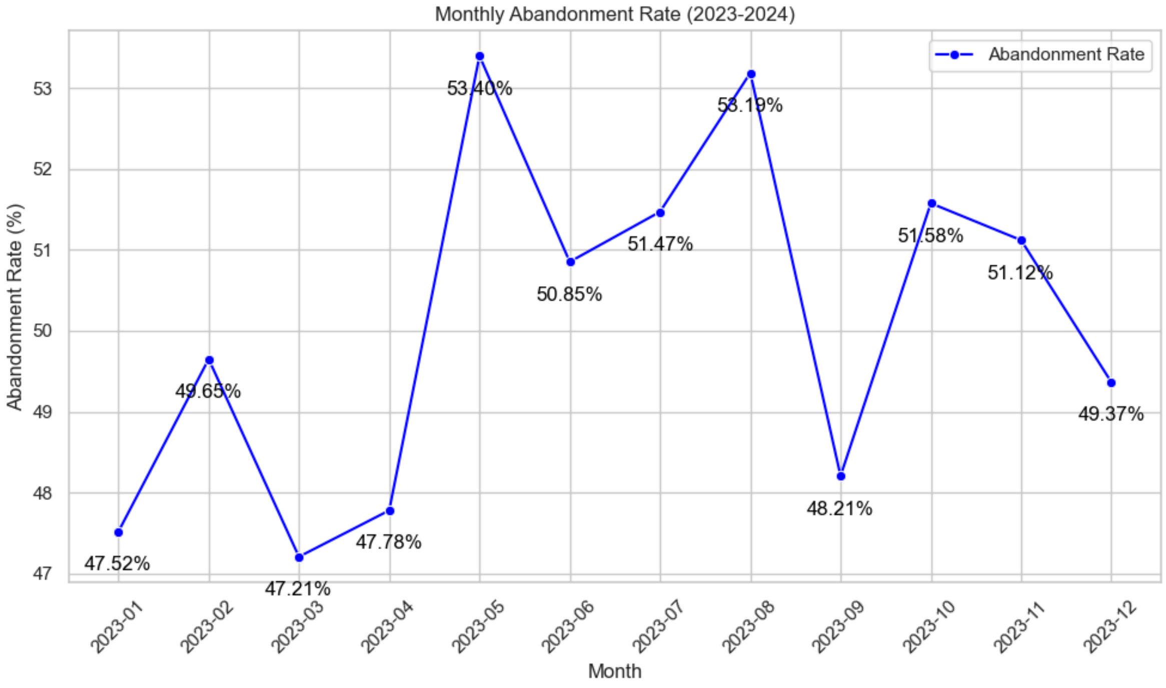
Shipping Costs/Time: Higher shipping costs or longer delivery times may lead to higher abandonment rates, especially in cities like London and Berlin. Currency & Payment Methods: Differences in payment options, local currency, and the availability of trusted payment methods could influence the purchase decision. Customer Behavior Differences: Customers in different cities may have varied expectations regarding product pricing, delivery speed, and overall convenience, contributing to differing abandonment rates. Actionable Insights:

Targeted Strategies for High Abandonment Cities:

For London and Berlin, focus on localizing the shopping experience by offering better shipping terms, providing more relevant product information, or implementing localized promotions to address the higher cart abandonment. Investigate the checkout process in these cities to identify any barriers that might be contributing to the high abandonment rates. Leverage New York's Lower Abandonment Rate: Study what factors contribute to New York's lower abandonment rate. This may provide insights into what aspects of the shopping experience are working well, which can be replicated in other cities.

In [52]:

```
# Cart Abandnomenr Rate Monthly Trend
monthly_abandnoment=pd.merge(date,customer_product_device_cart,how='left',on='da
monthly_abandnoment['month']=monthly_abandnoment['date'].dt.strftime('%Y-%m')
monthly_abandnomentrate=monthly_abandnoment.groupby(['month']).agg({'quantity':'
monthly_abandnomentrate['abandonment rate']=monthly_abandnomentrate['abandonment

# Set plot style
sns.set(style="whitegrid")

# Create the figure and axis
plt.figure(figsize=(10, 6))

# Plot the line chart
sns.lineplot(x='month', y='abandonment rate', data=monthly_abandnomentrate, mark

# Get y-axis limits to determine dynamic positioning of labels
y_min, y_max = plt.ylim()

# Add data labels for each point
for i, rate in enumerate(monthly_abandnomentrate['abandonment rate']):
    # Place the label above or below the point based on its value relative to th
    offset = 0.3 if rate < (y_max - y_min) / 2 else -0.3  # Adjust the offset dy
    plt.text(i, rate + offset, f'{rate:.2f}%', ha='center', va='bottom' if offse

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add labels and title
plt.ylabel('Abandonment Rate (%)')
plt.xlabel('Month')
plt.title('Monthly Abandonment Rate (2023-2024)')

# Adjust layout to prevent clipping
plt.tight_layout()

# Show plot
plt.show()
```



Key Observations and Insights:

Higher abandonment rates in May and August may be linked to factors such as seasonal shopping patterns, holiday sales, or promotions that increase traffic but also lead to higher cart abandonment. It could be useful to investigate if special events (e.g., sales, promotions) during these months contribute to this trend.

In months like May and August, where abandonment is particularly high, consider optimizing the checkout process, offering reminders, discount incentives, or limited-time offers to convert abandonments into completed purchases.

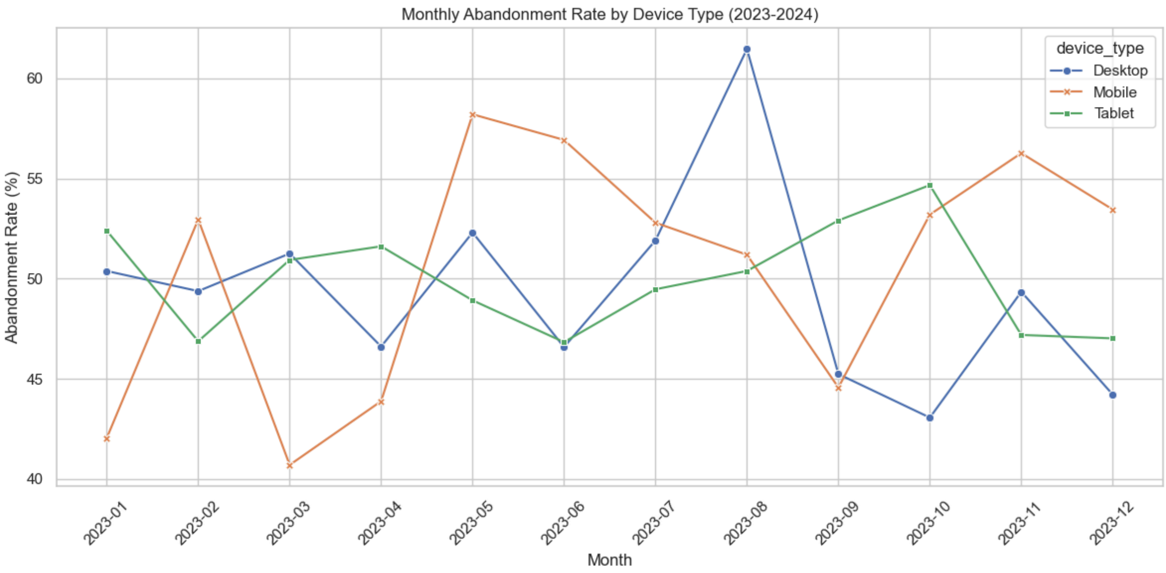
Actionable Insights:

For months like May and August with high abandonment rates, consider offering cart abandonment recovery tactics such as retargeting ads, special discounts, or fast checkout options to reduce hesitation and capture the customers who are likely to abandon their carts.

Investigate if special events, promotions, or seasonal factors (e.g., holidays or sales) in May and August are contributing to higher abandonment, and adjust strategies accordingly (e.g., offering incentives or improving product visibility).

Consider adjusting marketing strategies in months with higher abandonment rates to focus on conversion—for example, offering limited-time offers, free shipping, or easy returns to incentivize purchase completions.

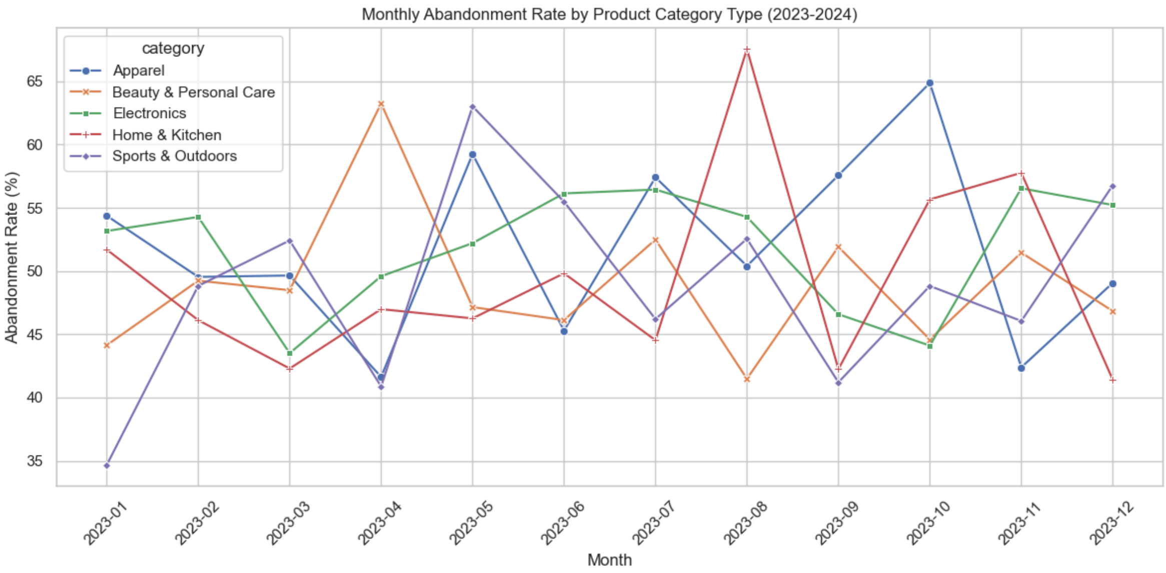
In [53]:
```
# Device specific Abandonment Rate
device_abandonment=monthly_abandnoment.groupby(['month','device_type']).agg({'qu
device_abandonment['abandonment rate']=device_abandonment['abandonment qty']*100

# Set plot style
sns.set(style="whitegrid")

# Create the figure and axis
plt.figure(figsize=(12, 6))

# Plot the line chart with Seaborn
sns.lineplot(x='month', y='abandonment rate', hue='device_type', style='device_t

# Add labels and title
plt.ylabel('Abandonment Rate (%)')
plt.xlabel('Month')
plt.title('Monthly Abandonment Rate by Device Type (2023-2024)')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add a grid for better visualization
plt.grid(True)

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



Key Observations Across Device Types:

Mobile Devices consistently have lower abandonment rates than tablets and desktops in most months, except for May, where both mobile and tablet abandonment rates are particularly high. Tablet and Desktop show higher abandonment rates in May, with tablet users also abandoning carts significantly across other months like July and October.

In [54]:
```
# Product Category wise Cart Abandonment
category_monthly=monthly_abandnoment.groupby(['month','category']).agg({'quantit
category_monthly['abandonment rate']=category_monthly['abandonment qty']*100/cat

# Set plot style
sns.set(style="whitegrid")

# Create the figure and axis
plt.figure(figsize=(12, 6))

# Plot the line chart with Seaborn
sns.lineplot(x='month', y='abandonment rate', hue='category', style='category',

# Add labels and title
plt.ylabel('Abandonment Rate (%)')
plt.xlabel('Month')
plt.title('Monthly Abandonment Rate by Product Category Type (2023-2024)')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add a grid for better visualization
plt.grid(True)

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



Observations:

Apparel Category:

The Apparel category consistently has high abandonment rates, with May seeing the highest rate at 59.25%, and October following closely with 64.86%. This suggests that customers might be experiencing issues such as sizing uncertainty, lack of detailed product information, or dissatisfaction with the overall purchasing process for apparel. Beauty & Personal Care:

The Beauty & Personal Care category has significant variability in abandonment rates across months, with April having the highest abandonment rate at 63.24%. This could indicate that customers may be unsure about product effectiveness, quality, or pricing, especially during certain months like April and July where abandonment spikes. Home & Kitchen:

The Home & Kitchen category saw a sharp spike in abandonment in August with a 67.54% rate, which is the highest for this category. This could point to issues like high costs or the complexity of decision-making for such products, especially in times of promotions or heavy sales periods. Recommendations:

Enhance Apparel Shopping Experience: Improve the sizing and return policy for the Apparel category. Consider implementing virtual try-on tools, detailed sizing guides, and free returns to mitigate the high abandonment rates. Additionally, providing discounts or limited-time offers might increase conversion, especially in months like May and October where abandonment is highest. Address Beauty & Personal Care Abandonment:

For the Beauty & Personal Care category, especially during months like April, it may be worth focusing on customer reviews, product trials, and clearer product descriptions. These improvements can help build trust, ensuring that customers feel confident about purchasing products. Offering sample packs or bundle deals could also increase conversions. Optimize the Home & Kitchen Category During High-Abandonment Months:

The Home & Kitchen category, particularly in August, has a high abandonment rate. To address this, consider offering price incentives, extended warranties, or free shipping on high-cost items. Additionally, reducing friction during checkout (e.g., guest checkouts or simplifying payment options) could help ease decision-making, especially during peak shopping periods.

Cohort Analysis

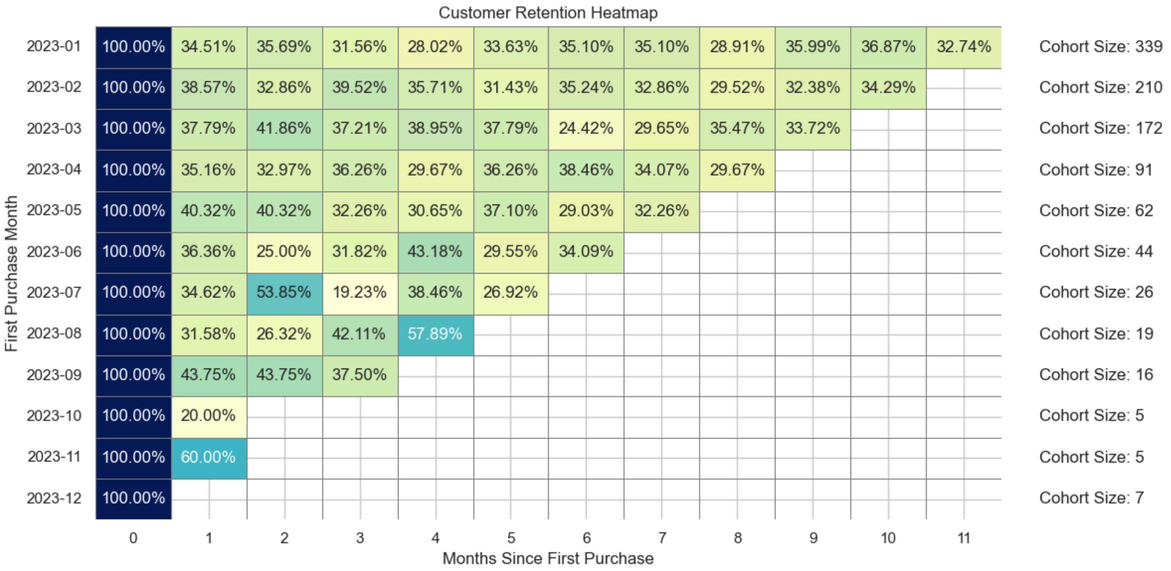
Customer Retention Cohort

In [55]:
```
monthly_abandnoment['Order_Month'] = monthly_abandnoment['date'].dt.to_period('M
monthly_abandnoment['Cohort']=monthly_abandnoment.groupby(['customer_id'])['date
df_ch1=monthly_abandnoment.groupby(['Order_Month','Cohort']).agg({'customer_id':
```

In [56]:
```
from operator import attrgetter
df_ch1['Cohort_Periods'] = (df_ch1.Order_Month - df_ch1.Cohort).apply(attrgetter
cohort_pivot=df_ch1.pivot(index='Cohort',columns='Cohort_Periods',values='unique
```

In [57]:
```
retention_percentage = cohort_pivot.divide(cohort_pivot.iloc[:, 0], axis=0)
cohort_size = cohort_pivot.iloc[:, 0]
```

In [58]:
```
plt.figure(figsize=(12, 6))

# Plot the heatmap
sns.heatmap(data=retention_percentage, annot=True, cmap='YlGnBu', fmt='.2%', lin
plt.title('Customer Retention Heatmap')
plt.xlabel('Months Since First Purchase')
plt.ylabel('First Purchase Month')

# Add cohort size as annotations
for i in range(len(cohort_size)):
    plt.text(len(cohort_size) + 0.5, i + 0.5, f'Cohort Size: {int(cohort_size.il

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



Observations

Significant Decline in Retention Over Time:

Retention rates consistently decline as customers move further from the initial month (Month 0). For example, in 2023-01, retention drops from 100% in the first month to 32.74% by Month 11. This indicates that customers are more likely to disengage over time, with long-term retention becoming a key challenge.

Higher Retention in Early Months for Some Cohorts:

Certain cohorts, such as 2023-11, exhibit higher retention in the first few months, with 60% retention in Month 1. This suggests that specific strategies implemented in these periods may have led to better initial customer engagement, which could be leveraged further to boost long-term retention.

Inconsistent Retention Across Cohorts:

There is significant variability in retention across cohorts. For example:The 2023-05 cohort has a high initial retention but experiences a steep decline afterward.The 2023-08 cohort shows a notable spike in Month 4 (57.89%), but the pattern is inconsistent compared to other cohorts.2023-10 and 2023-11 cohorts have fewer data points, reflecting potential incomplete or newer cohort data.

Recommendations to Improve Retention:

Implement Proactive Engagement Strategies Early On:

Given that retention rates decline over time, focus on maintaining customer interest during the critical months after the initial purchase. Consider offering loyalty rewards, personalized follow-ups, or special promotions in months 1 to 6 to re-engage customers and boost long-term retention. This could be particularly useful for cohorts like 2023-01 and 2023-05, which show significant drop-offs after the first few months

Replicate Strategies from Higher Retention Cohorts:

Cohorts like 2023-11 show strong early retention. Analyze the factors that contributed to the higher retention in Month 1 (e.g., targeted campaigns, product satisfaction, or loyalty incentives) and apply similar strategies to other cohorts, particularly those with high drop-offs (e.g., 2023-05 or 2023-10). This could involve personalized onboarding, product education, or early incentives to encourage repeat purchases.

Cart Anandnoment Rate Cohort

In [62]:

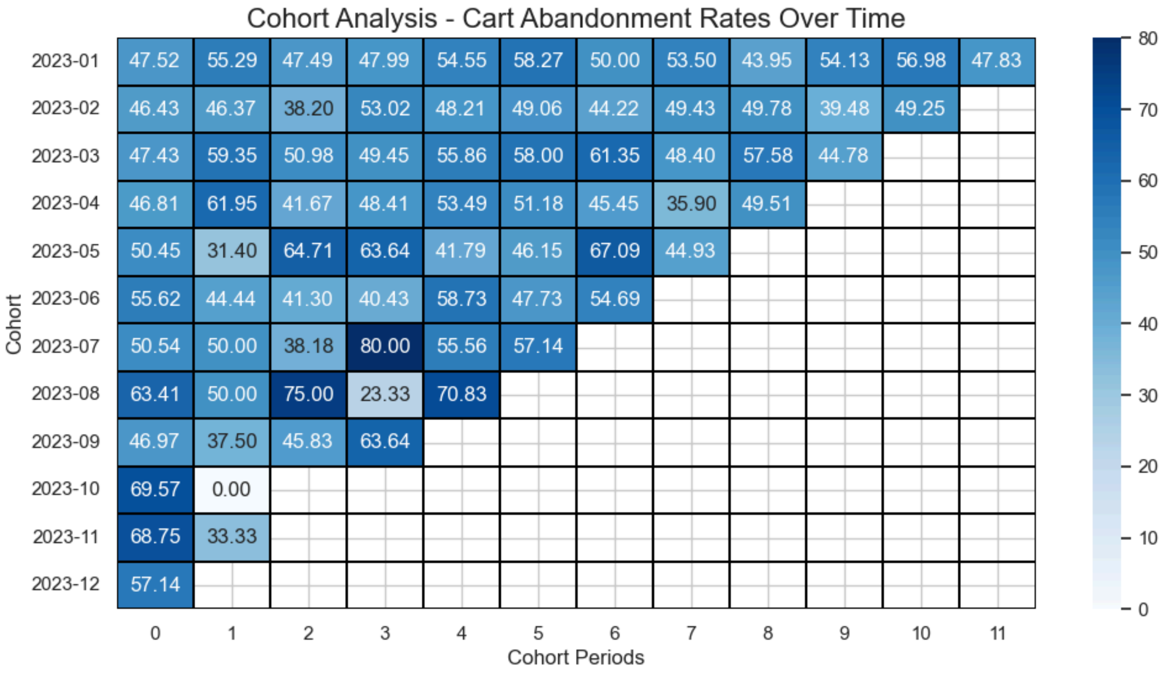
```
monthlt_abandnoment_rate=monthly_abandnoment.groupby(['Order_Month','Cohort']).a
monthlt_abandnoment_rate['abandonment rate']=monthlt_abandnoment_rate['abandonme
monthlt_abandnoment_rate['abandonment rate']=monthlt_abandnoment_rate['abandonme
df_ch2=monthlt_abandnoment_rate
```

In [63]:

```
from operator import attrgetter
df_ch2['Cohort_Periods'] = (df_ch2.Order_Month - df_ch2.Cohort).apply(attrgetter
abandonment_cohort_pivot=df_ch2.pivot(index='Cohort',columns='Cohort_Periods',va
abandonment_cohort_pivot=abandonment_cohort_pivot.reset_index()
```

In [64]:

```
# Set 'Cohort' as the index
abandonment_cohort_pivot.set_index('Cohort', inplace=True)

# Set plot size
plt.figure(figsize=(12, 6))

# Create the heatmap
sns.heatmap(abandonment_cohort_pivot, annot=True, fmt=".2f", cmap="Blues", linew

# Add title and labels
plt.title('Cohort Analysis - Cart Abandonment Rates Over Time', fontsize=16)
plt.xlabel('Cohort Periods', fontsize=12)
plt.ylabel('Cohort', fontsize=12)

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



Observations

High Fluctuations in Abandonment Rates Over Time:

In some cohorts, such as 2023-05 and 2023-07, there are significant fluctuations in abandonment rates across periods. For instance: In 2023-05, abandonment rates jump from 31.40% in Month 1 to 64.71% in Month 2, and then peak at 67.09% in Month 6, showing high volatility. This suggests that a large number of customers from this cohort may be abandoning their carts over a short span. Similarly, 2023-07 shows a sharp peak at 80.00% in Month 3. Such high variability indicates a need for deeper analysis of the factors driving these changes.

Month 0 Retention vs. Long-Term Abandonment:

The Month 0 (initial period) shows relatively higher retention, but as the months progress, there is a noticeable decline in retention, resulting in increasing abandonment rates. For example: In 2023-01, the abandonment rate is around 47.52% in Month 0, but it rises gradually to 56.98% in Month 10. This steady increase in abandonment suggests that retention strategies may not be as effective as time progresses.

Recommendations to Improve Customer Retention and Reduce Abandonment:

Analyze Causes of High Fluctuation in Abandonment Rates:

For cohorts like 2023-05 and 2023-07, where there are sharp increases in abandonment rates in specific months (e.g., Month 2 and Month 3), conduct further analysis to determine the exact causes. It could be related to: Seasonal promotions, shipping delays, or pricing changes in these months. A suboptimal checkout experience, which could be driving higher abandonment during these periods. Once these factors are identified, tailor retention strategies to mitigate abandonment in these critical months.

Strengthen Long-Term Retention Strategies:

As customer abandonment rates increase over time, it's essential to introduce retention initiatives that extend beyond the initial period. Some potential strategies could include: Offering post-purchase follow-ups like discounts or loyalty points for repeat purchases. Sending personalized reminders or re-engagement emails targeting customers who have abandoned carts after a certain period. Implementing time-limited offers to encourage return visits and purchases during periods of high abandonment (e.g., Month 2 to Month 6).

In [66]:

```
cities=monthly_abandnoment['city'].unique()
cities
```

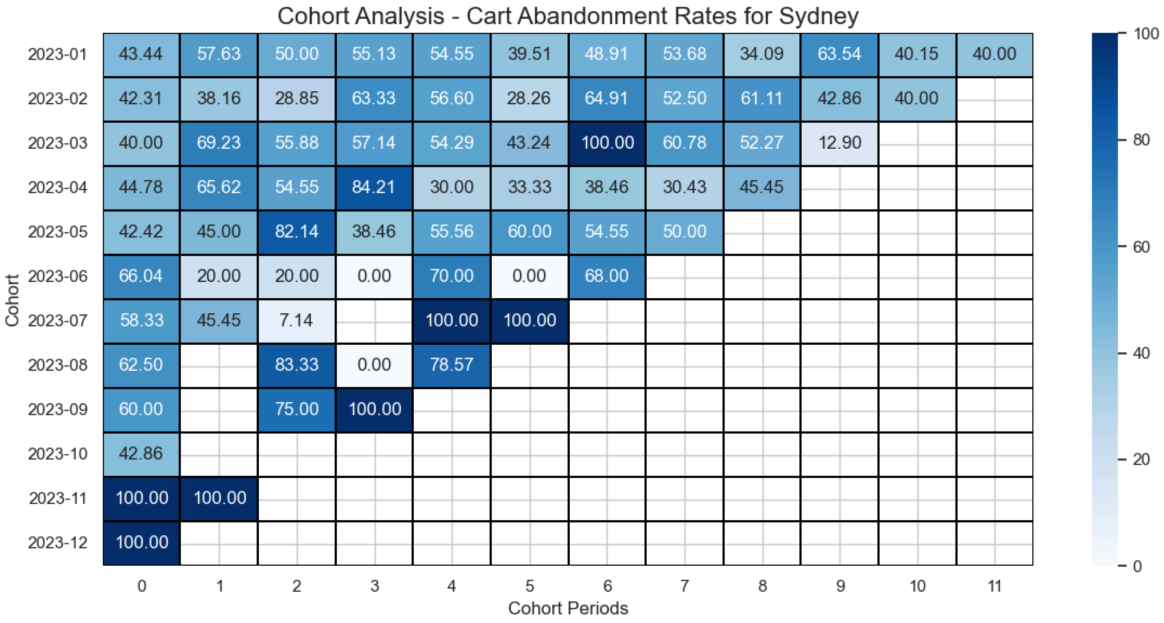
Out[66]:

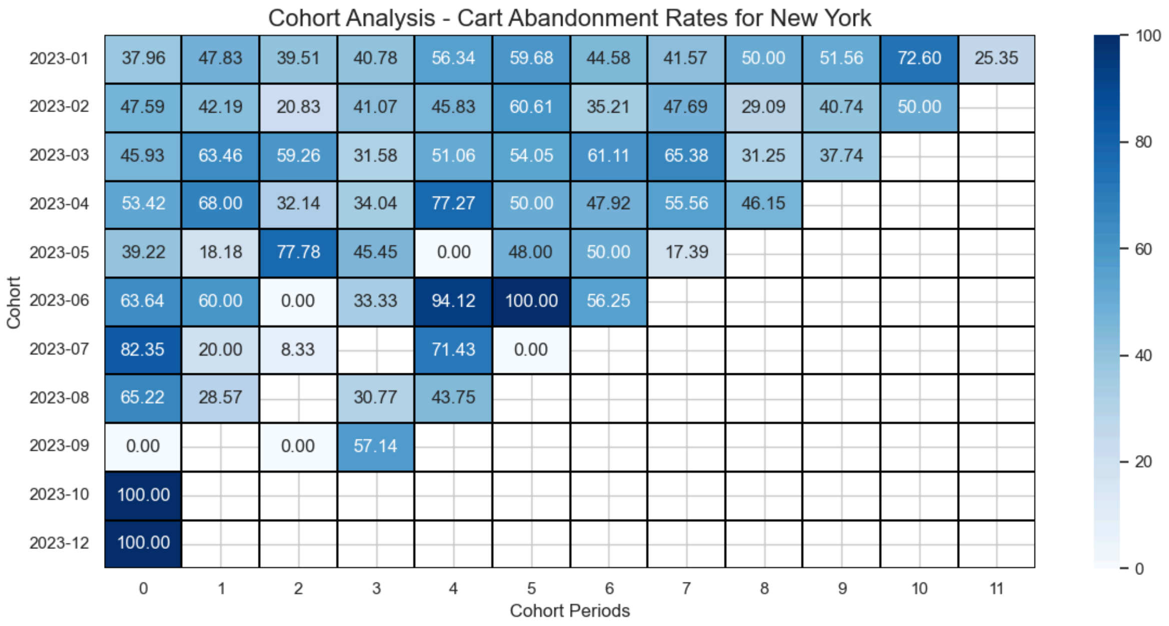
```
array(['Sydney', 'New York', 'Mumbai', 'Berlin', 'London'], dtype=object)
```

Citywise Cart Abandonment Rate

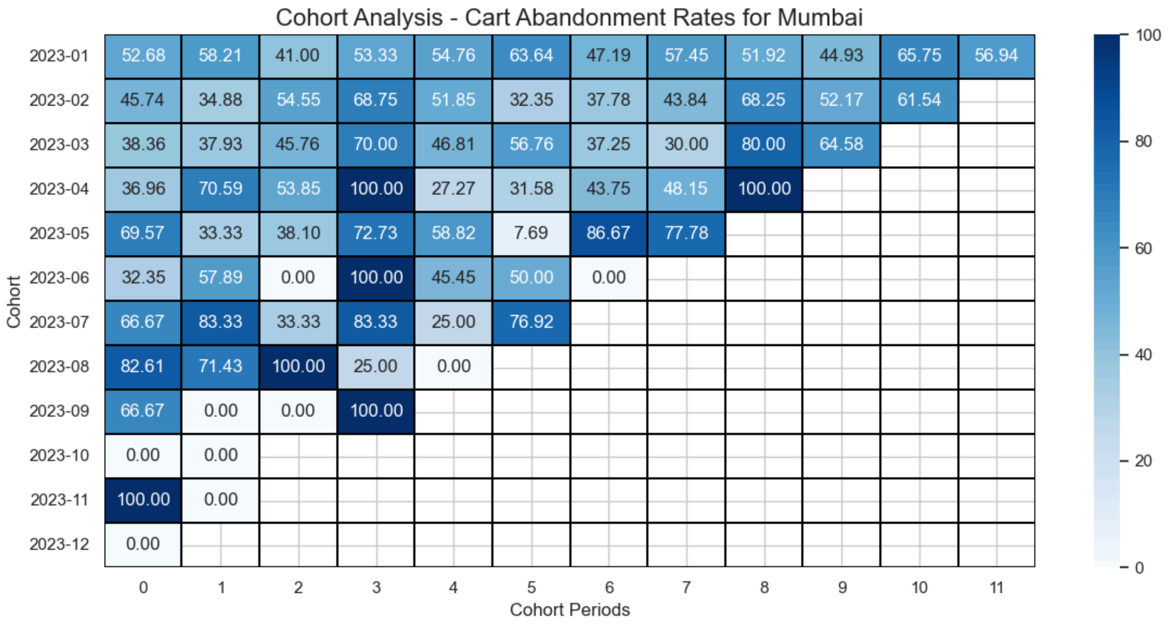
In [69]:

```
for i in cities:
    # Filter data by city
    city_cohort = monthly_abandnoment[monthly_abandnoment['city'] == i]

    # Group by 'Order_Month' and 'Cohort', aggregating quantity and abandonment
    city_cohort1 = city_cohort.groupby(['Order_Month', 'Cohort']).agg({'quantity
    # Calculate abandonment rate
    city_cohort1['abandonment rate'] = (city_cohort1['abandonment qty'] / city_c
    city_cohort1['abandonment rate'] = city_cohort1['abandonment rate'].round(2)

    # Calculate Cohort_Periods (difference in months between Order_Month and Coh
    city_cohort1['Cohort_Periods'] = (city_cohort1['Order_Month'] - city_cohort1

    # Pivot data to create cohort heatmap structure
    city_cohort1_pivot = city_cohort1.pivot(index='Cohort', columns='Cohort_Peri

    # Set 'Cohort' as the index
    city_cohort1_pivot.reset_index(inplace=True)
    city_cohort1_pivot.set_index('Cohort', inplace=True)

    # Set plot size
    plt.figure(figsize=(12, 6))

    # Create the heatmap
    sns.heatmap(city_cohort1_pivot, annot=True, fmt=".2f", cmap="Blues", linewid

    # Add title and labels
    plt.title(f'Cohort Analysis - Cart Abandonment Rates for {i}', fontsize=16)
    plt.xlabel('Cohort Periods', fontsize=12)
    plt.ylabel('Cohort', fontsize=12)

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

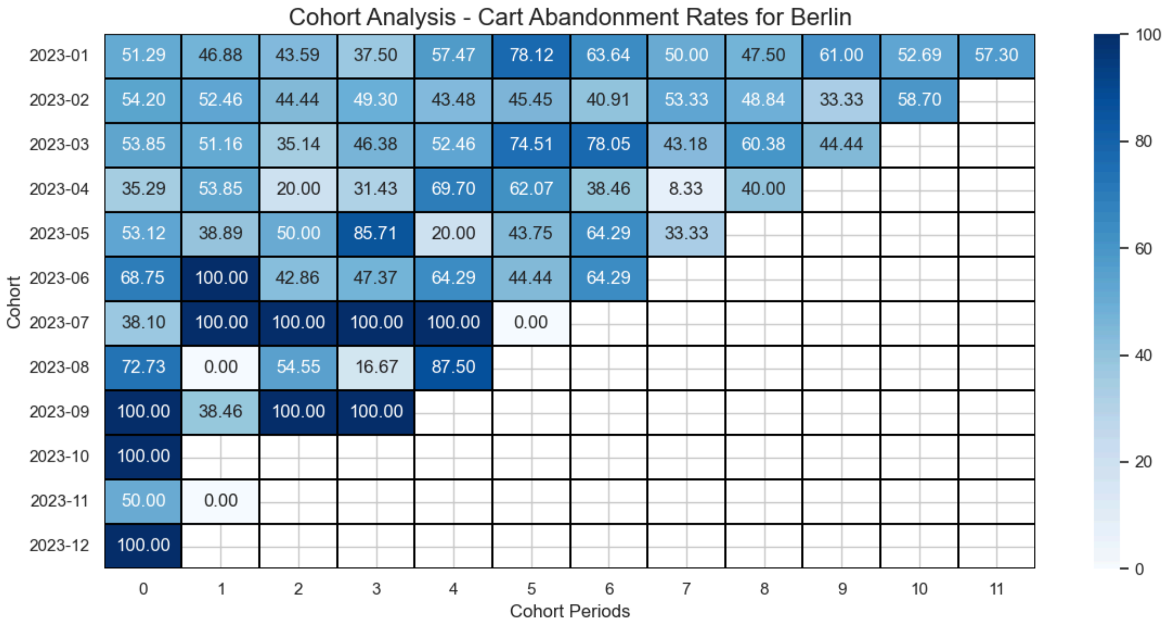




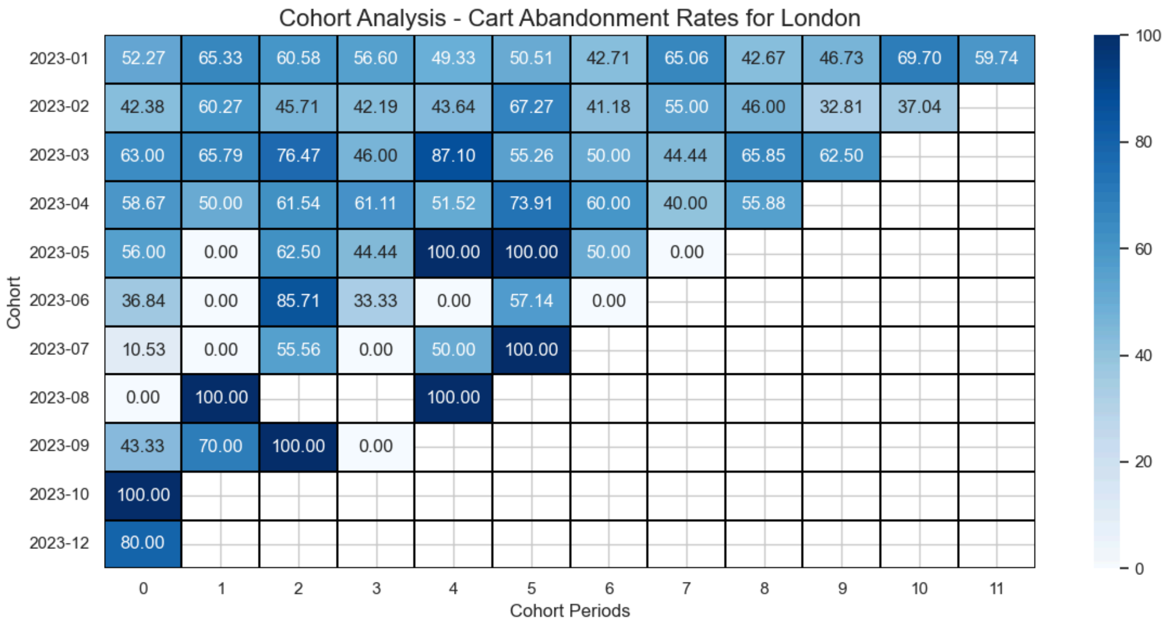
Cohort	0	1	2	3	4	5	6	7	8	9	10	11
2023-01	37.96	47.83	39.51	40.78	56.34	59.68	44.58	41.57	50.00	51.56	72.60	25.35
2023-02	47.59	42.19	20.83	41.07	45.83	60.61	35.21	47.69	29.09	40.74	50.00	
2023-03	45.93	63.46	59.26	31.58	51.06	54.05	61.11	65.38	31.25	37.74		
2023-04	53.42	68.00	32.14	34.04	77.27	50.00	47.92	55.56	46.15			
2023-05	39.22	18.18	77.78	45.45	0.00	48.00	50.00	17.39				
2023-06	63.64	60.00	0.00	33.33	94.12	100.00	56.25					
2023-07	82.35	20.00	8.33		71.43	0.00						
2023-08	65.22	28.57		30.77	43.75							
2023-09	0.00		0.00	57.14								
2023-10	100.00											
2023-12	100.00											



Cohort	0	1	2	3	4	5	6	7	8	9	10	11
2023-01	52.68	58.21	41.00	53.33	54.76	63.64	47.19	57.45	51.92	44.93	65.75	56.94
2023-02	45.74	34.88	54.55	68.75	51.85	32.35	37.78	43.84	68.25	52.17	61.54	
2023-03	38.36	37.93	45.76	70.00	46.81	56.76	37.25	30.00	80.00	64.58		
2023-04	36.96	70.59	53.85	100.00	27.27	31.58	43.75	48.15	100.00			
2023-05	69.57	33.33	38.10	72.73	58.82	7.69	86.67	77.78				
2023-06	32.35	57.89	0.00	100.00	45.45	50.00	0.00					
2023-07	66.67	83.33	33.33	83.33	25.00	76.92						
2023-08	82.61	71.43	100.00	25.00	0.00							
2023-09	66.67	0.00	0.00	100.00								
2023-10	0.00	0.00										
2023-11	100.00	0.00										
2023-12	0.00											

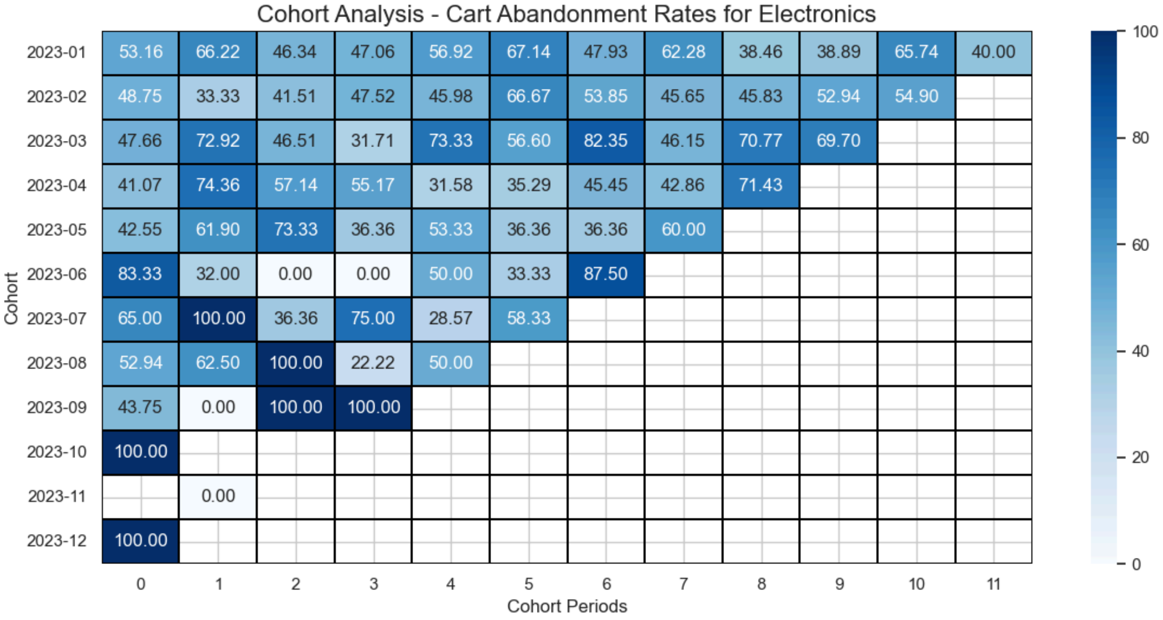


Cohort	0	1	2	3	4	5	6	7	8	9	10	11
2023-01	51.29	46.88	43.59	37.50	57.47	78.12	63.64	50.00	47.50	61.00	52.69	57.30
2023-02	54.20	52.46	44.44	49.30	43.48	45.45	40.91	53.33	48.84	33.33	58.70	
2023-03	53.85	51.16	35.14	46.38	52.46	74.51	78.05	43.18	60.38	44.44		
2023-04	35.29	53.85	20.00	31.43	69.70	62.07	38.46	8.33	40.00			
2023-05	53.12	38.89	50.00	85.71	20.00	43.75	64.29	33.33				
2023-06	68.75	100.00	42.86	47.37	64.29	44.44	64.29					
2023-07	38.10	100.00	100.00	100.00	100.00	0.00						
2023-08	72.73	0.00	54.55	16.67	87.50							
2023-09	100.00	38.46	100.00	100.00								
2023-10	100.00											
2023-11	50.00	0.00										
2023-12	100.00											



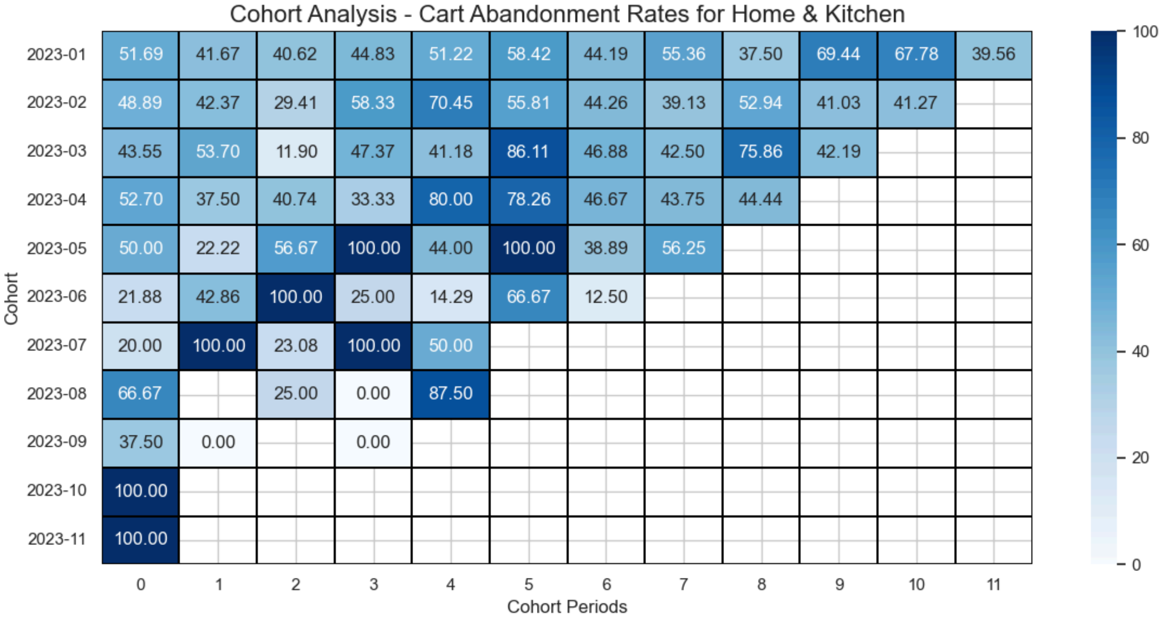
Product Categorywise Cart Abandonment Cohort

In [72]:
```
category=monthly_abandnoment['category'].unique()
for i in category:
    # Filter data by city
    category_cohort = monthly_abandnoment[monthly_abandnoment['category'] == i]

    # Group by 'Order_Month' and 'Cohort', aggregating quantity and abandonment
    category_cohort1 = category_cohort.groupby(['Order_Month', 'Cohort']).agg({'

    # Calculate abandonment rate
    category_cohort1['abandonment rate'] = (category_cohort1['abandonment qty']
    category_cohort1['abandonment rate'] = category_cohort1['abandonment rate'].

# Calculate Cohort_Periods (difference in months between Order_Month and Cohort)
    category_cohort1['Cohort_Periods'] = (category_cohort1['Order_Month'] - cate

    # Pivot data to create cohort heatmap structure
    category_cohort1_pivot = category_cohort1.pivot(index='Cohort', columns='Coh

    # Set 'Cohort' as the index
    category_cohort1_pivot.reset_index(inplace=True)
    category_cohort1_pivot.set_index('Cohort', inplace=True)

    # Set plot size
    plt.figure(figsize=(12, 6))

    # Create the heatmap
    sns.heatmap(category_cohort1_pivot, annot=True, fmt=".2f", cmap="Blues", lin

    # Add title and labels
    plt.title(f'Cohort Analysis - Cart Abandonment Rates for {i}', fontsize=16)
    plt.xlabel('Cohort Periods', fontsize=12)
    plt.ylabel('Cohort', fontsize=12)

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



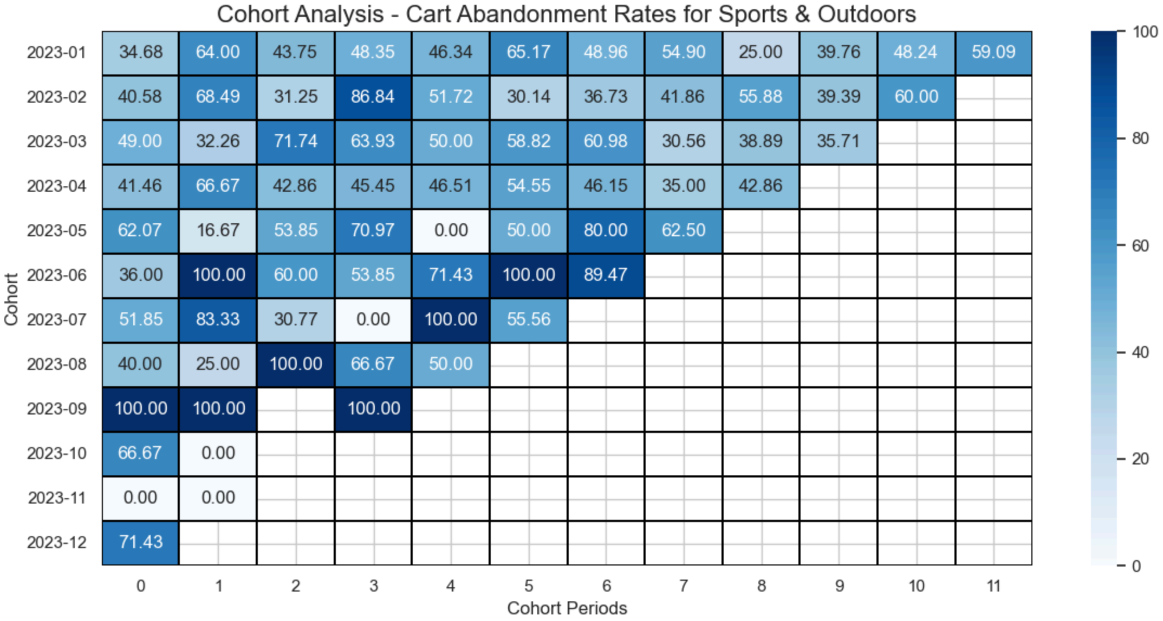
Cohort Analysis - Cart Abandonment Rates for Electronics

Cohort	0	1	2	3	4	5	6	7	8	9	10	11
2023-01	53.16	66.22	46.34	47.06	56.92	67.14	47.93	62.28	38.46	38.89	65.74	40.00
2023-02	48.75	33.33	41.51	47.52	45.98	66.67	53.85	45.65	45.83	52.94	54.90	
2023-03	47.66	72.92	46.51	31.71	73.33	56.60	82.35	46.15	70.77	69.70		
2023-04	41.07	74.36	57.14	55.17	31.58	35.29	45.45	42.86	71.43			
2023-05	42.55	61.90	73.33	36.36	53.33	36.36	36.36	60.00				
2023-06	83.33	32.00	0.00	0.00	50.00	33.33	87.50					
2023-07	65.00	100.00	36.36	75.00	28.57	58.33						
2023-08	52.94	62.50	100.00	22.22	50.00							
2023-09	43.75	0.00	100.00	100.00								
2023-10	100.00											
2023-11		0.00										
2023-12	100.00											



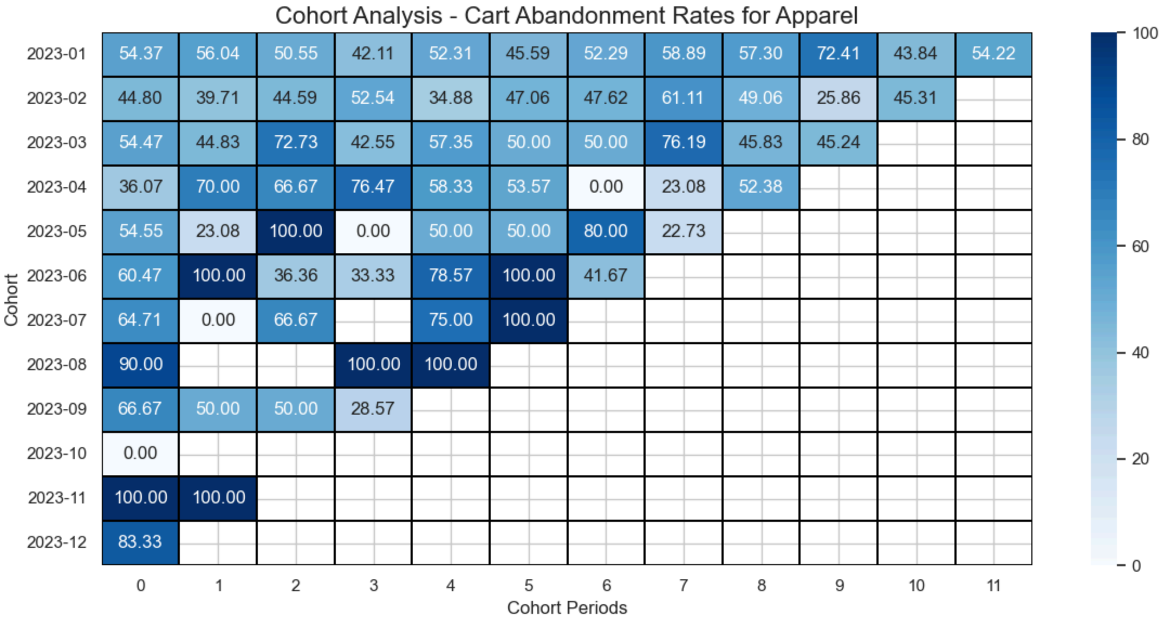
Cohort Analysis - Cart Abandonment Rates for Home & Kitchen

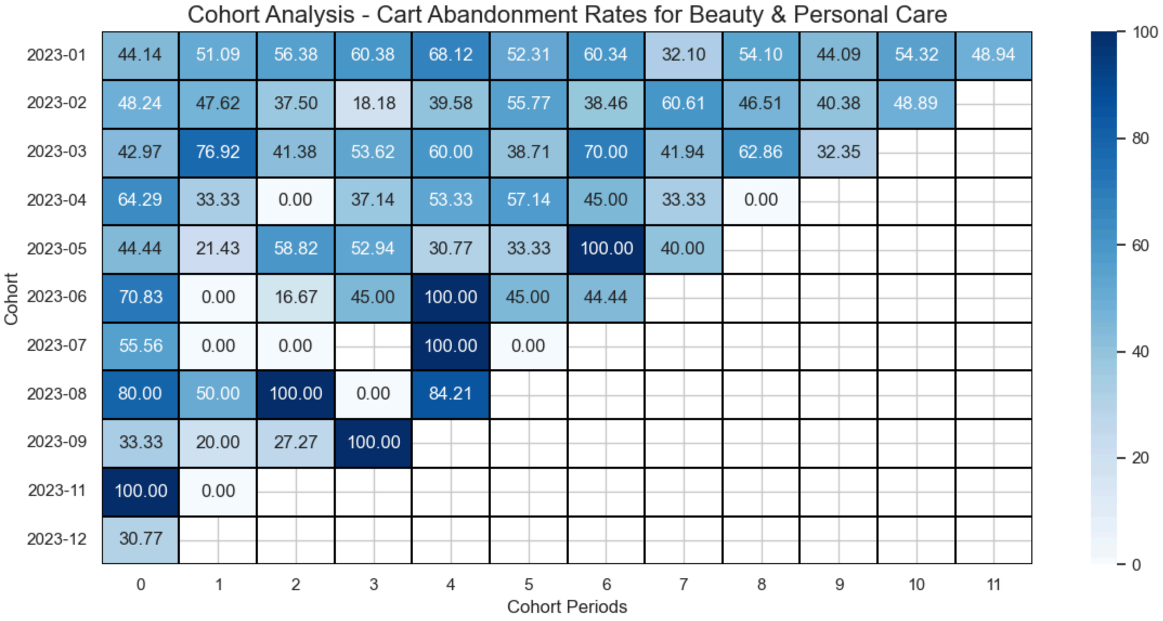
Cohort	0	1	2	3	4	5	6	7	8	9	10	11
2023-01	51.69	41.67	40.62	44.83	51.22	58.42	44.19	55.36	37.50	69.44	67.78	39.56
2023-02	48.89	42.37	29.41	58.33	70.45	55.81	44.26	39.13	52.94	41.03	41.27	
2023-03	43.55	53.70	11.90	47.37	41.18	86.11	46.88	42.50	75.86	42.19		
2023-04	52.70	37.50	40.74	33.33	80.00	78.26	46.67	43.75	44.44			
2023-05	50.00	22.22	56.67	100.00	44.00	100.00	38.89	56.25				
2023-06	21.88	42.86	100.00	25.00	14.29	66.67	12.50					
2023-07	20.00	100.00	23.08	100.00	50.00							
2023-08	66.67		25.00	0.00	87.50							
2023-09	37.50	0.00		0.00								
2023-10	100.00											
2023-11	100.00											



Cohort Analysis - Cart Abandonment Rates for Sports & Outdoors

Cohort	0	1	2	3	4	5	6	7	8	9	10	11
2023-01	34.68	64.00	43.75	48.35	46.34	65.17	48.96	54.90	25.00	39.76	48.24	59.09
2023-02	40.58	68.49	31.25	86.84	51.72	30.14	36.73	41.86	55.88	39.39	60.00	
2023-03	49.00	32.26	71.74	63.93	50.00	58.82	60.98	30.56	38.89	35.71		
2023-04	41.46	66.67	42.86	45.45	46.51	54.55	46.15	35.00	42.86			
2023-05	62.07	16.67	53.85	70.97	0.00	50.00	80.00	62.50				
2023-06	36.00	100.00	60.00	53.85	71.43	100.00	89.47					
2023-07	51.85	83.33	30.77	0.00	100.00	55.56						
2023-08	40.00	25.00	100.00	66.67	50.00							
2023-09	100.00	100.00		100.00								
2023-10	66.67	0.00										
2023-11	0.00	0.00										
2023-12	71.43											

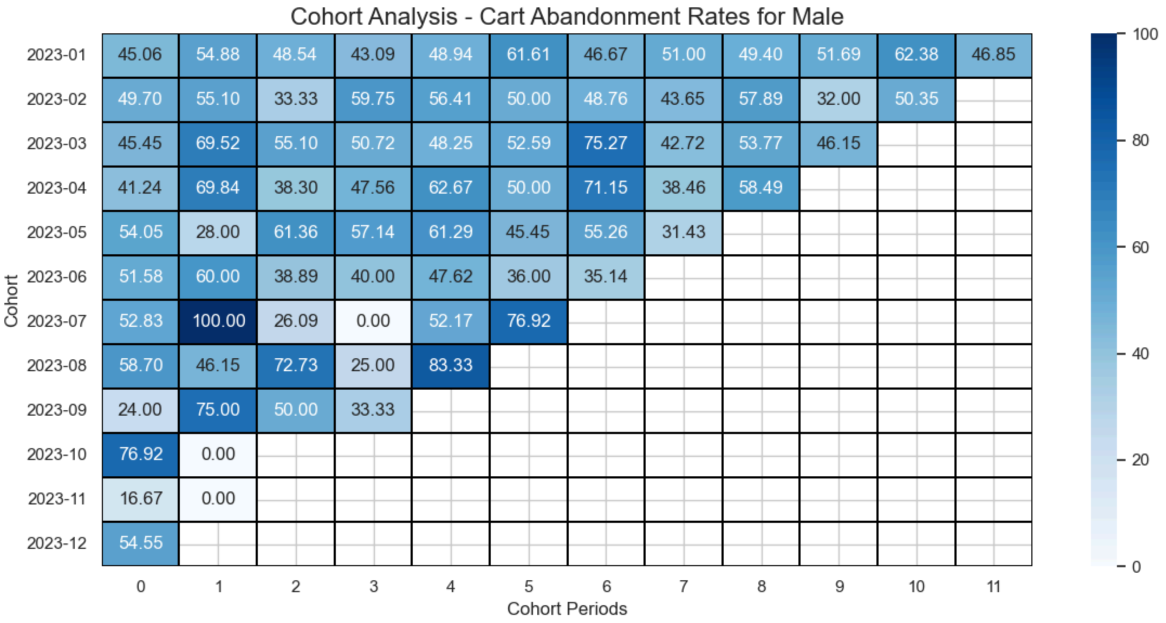

Cohort Analysis - Cart Abandonment Rates for Apparel

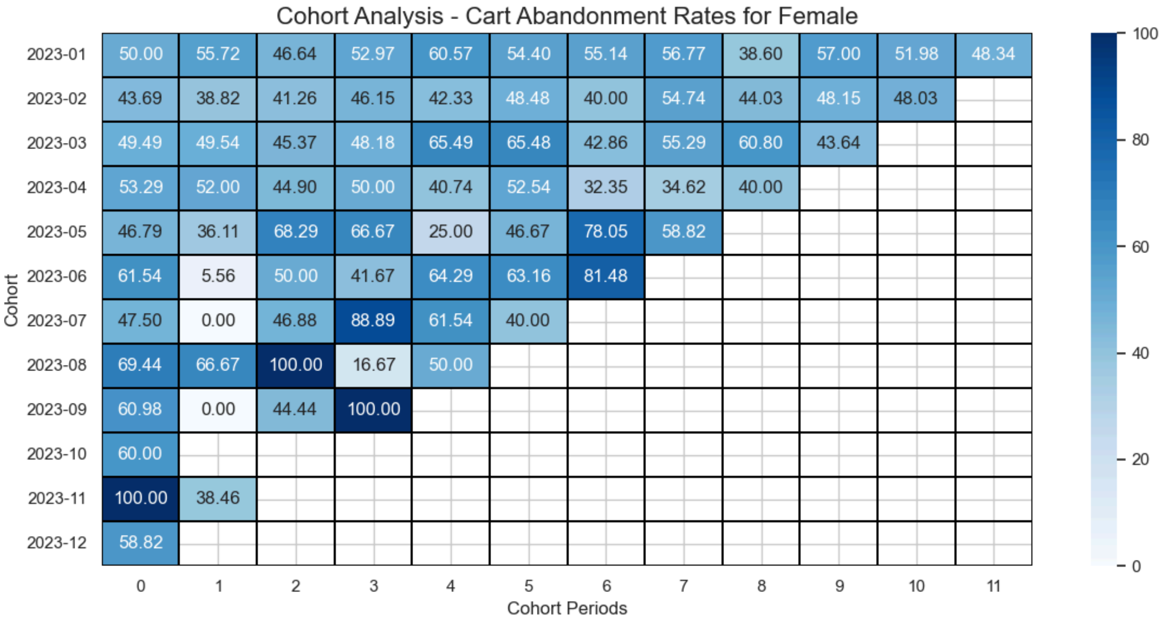

Cohort Analysis - Cart Abandonment Rates for Beauty & Personal Care

Gender Wise Cart Abandonment Cohort

In [73]:

```
gender=monthly_abandnoment['gender'].unique()
for i in gender:
    # Filter data by city
    gender_cohort = monthly_abandnoment[monthly_abandnoment['gender'] == i]

    # Group by 'Order_Month' and 'Cohort', aggregating quantity and abandonment
    gender_cohort1 = gender_cohort.groupby(['Order_Month', 'Cohort']).agg({'quan

    # Calculate abandonment rate
    gender_cohort1['abandonment rate'] = (gender_cohort1['abandonment qty'] / ge
    gender_cohort1['abandonment rate'] = gender_cohort1['abandonment rate'].roun

    # Calculate Cohort_Periods (difference in months between Order_Month and Coh
    gender_cohort1['Cohort_Periods'] = (gender_cohort1['Order_Month'] - gender_c

    # Pivot data to create cohort heatmap structure
    gender_cohort1_pivot = gender_cohort1.pivot(index='Cohort', columns='Cohort_
# Set 'Cohort' as the index
    gender_cohort1_pivot.reset_index(inplace=True)
    gender_cohort1_pivot.set_index('Cohort', inplace=True)

    # Set plot size
    plt.figure(figsize=(12, 6))

    # Create the heatmap
    sns.heatmap(gender_cohort1_pivot, annot=True, fmt=".2f", cmap="Blues", linew

    # Add title and labels
    plt.title(f'Cohort Analysis - Cart Abandonment Rates for {i}', fontsize=16)
    plt.xlabel('Cohort Periods', fontsize=12)
    plt.ylabel('Cohort', fontsize=12)

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





In [ ]: