Innobyte Services Internship Task

Name : PALLAVI PATIL

Amazon Sales Analysis

Objective : To Analyze and Provide Insights on Amazon Sales Report

Libraries

In [47]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import calendar
```

Dataset

In [3]:
```
df = pd.read_csv("C:\\Users\\Dell\\Downloads\\Amazon_sales.csv")
```

1.Shape

In [4]:
```
df.shape
```

Out[4]: (128976, 21)

In [5]:
```
df.head(3)
```

Out[5]:

	index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Category	Size	Courier Status	...	currency	Amount
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	T-shirt	S	On the Way	...	INR	647.62
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	Shirt	3XL	Shipped	...	INR	406.00
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	Shirt	XL	Shipped	...	INR	329.00

3 rows × 21 columns

In [6]:
```
1 df.tail(3)
```

Out[6]:

	index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Category	Size	Courier Status	...	currency	Amo
128973	128972	407-9547469-3152358	05-31-22	Shipped	Amazon	Amazon.in	Expedited	Blazzer	XXL	Shipped	...	INR	69
128974	128973	402-6184140-0545956	05-31-22	Shipped	Amazon	Amazon.in	Expedited	T-shirt	XS	Shipped	...	INR	119
128975	128974	408-7436540-8728312	05-31-22	Shipped	Amazon	Amazon.in	Expedited	T-shirt	S	Shipped	...	INR	69

3 rows × 21 columns

2. Info

In [7]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128976 entries, 0 to 128975
Data columns (total 21 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   index               128976 non-null  int64  
 1   Order ID            128976 non-null  object 
 2   Date                128976 non-null  object 
 3   Status              128976 non-null  object 
 4   Fulfilment          128976 non-null  object 
 5   Sales Channel       128976 non-null  object 
 6   ship-service-level  128976 non-null  object 
 7   Category            128976 non-null  object 
 8   Size                128976 non-null  object 
 9   Courier Status      128976 non-null  object 
 10  Qty                 128976 non-null  int64  
 11  currency            121176 non-null  object 
 12  Amount              121176 non-null  float64
 13  ship-city           128941 non-null  object 
 14  ship-state          128941 non-null  object 
 15  ship-postal-code    128941 non-null  float64
 16  ship-country        128941 non-null  object 
 17  B2B                 128976 non-null  bool   
 18  fulfilled-by        39263 non-null   object 
 19  New                 0 non-null       float64
 20  PendingS            0 non-null       float64
dtypes: bool(1), float64(4), int64(2), object(14)
memory usage: 19.8+ MB
```

3. Describe

In [8]:
```
1 df.describe()
```

Out[8]:

	index	Qty	Amount	ship-postal-code	New	PendingS
count	128976.000000	128976.000000	121176.000000	128941.000000	0.0	0.0
mean	64486.130427	0.904401	648.562176	463945.677744	NaN	NaN
std	37232.897832	0.313368	281.185041	191458.488954	NaN	NaN
min	0.000000	0.000000	0.000000	110001.000000	NaN	NaN
25%	32242.750000	1.000000	449.000000	382421.000000	NaN	NaN
50%	64486.500000	1.000000	605.000000	500033.000000	NaN	NaN
75%	96730.250000	1.000000	788.000000	600024.000000	NaN	NaN
max	128974.000000	15.000000	5584.000000	989898.000000	NaN	NaN

4. Missing Values

In [9]:
```
1 df.isna().sum()
```

Out[9]:
```
index                   0
Order ID                0
Date                    0
Status                  0
Fulfilment              0
Sales Channel           0
ship-service-level      0
Category                0
Size                    0
Courier Status          0
Qty                     0
currency             7800
Amount               7800
ship-city              35
ship-state             35
ship-postal-code       35
ship-country           35
B2B                     0
fulfilled-by        89713
New                128976
PendingS           128976
dtype: int64
```

Handling missing values

In [10]:
```
1 df1=df.drop(columns=["New","PendingS","index"])
```

In [11]:
```
1 df1[df1.duplicated()].shape
```

Out[11]:
```
(959, 18)
```

In [12]:
```
1 df1.drop_duplicates(inplace=True)
```

In [13]:
```
1 df1 = df1.loc[df1["Qty"] != 0]
```

In [14]:
```
1 df1.isna().sum()
```

Out[14]:
```
Order ID                0
Date                    0
Status                  0
Fulfilment              0
Sales Channel           0
ship-service-level      0
Category                0
Size                    0
Courier Status          0
Qty                     0
currency              124
Amount                124
ship-city              27
ship-state             27
ship-postal-code       27
ship-country           27
B2B                     0
fulfilled-by        83341
dtype: int64
```

In [15]:
```
1 df1.dropna(subset=["Amount","currency"],inplace=True)
```

In [16]:
```
1 df1.dropna(subset=["ship-city","ship-state","ship-postal-code","ship-country"],inplace=True
```

In [17]:
```
1 df1.isna().sum()
```

Out[17]:
```
Order ID                0
Date                    0
Status                  0
Fulfilment              0
Sales Channel           0
ship-service-level      0
Category                0
Size                    0
Courier Status          0
Qty                     0
currency                0
Amount                  0
ship-city               0
ship-state              0
ship-postal-code        0
ship-country            0
B2B                     0
fulfilled-by        83200
dtype: int64
```

In [18]:
```
1 df1['fulfilled-by'].fillna('Unknown', inplace=True)
```

In [19]:
```
df1.isna().sum()
```

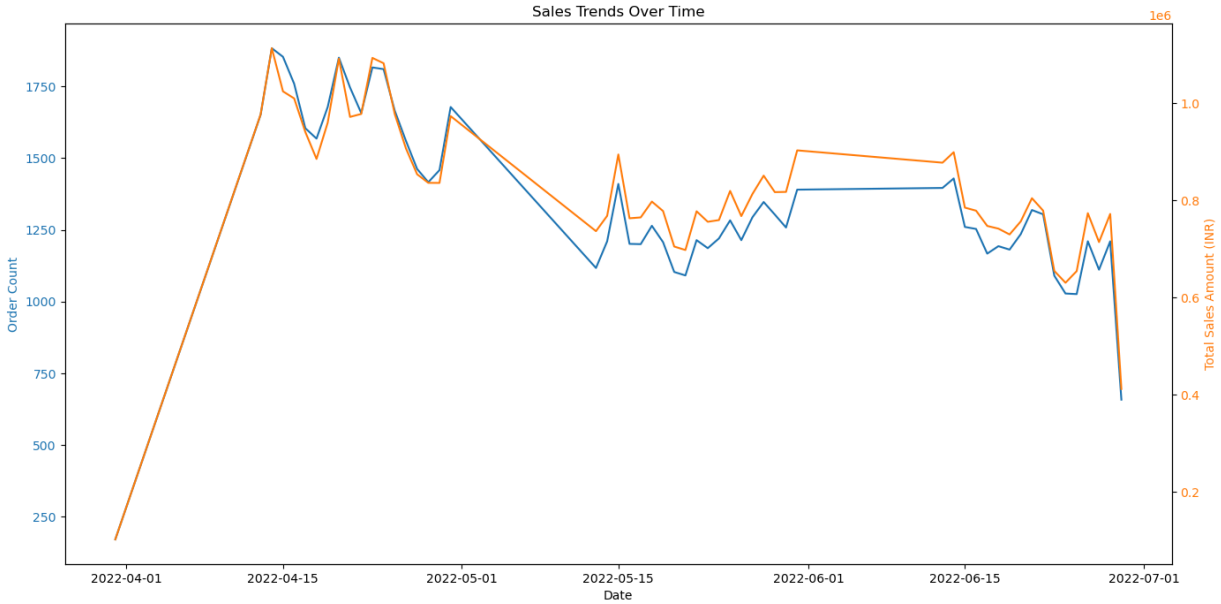
Out[19]:
```
Order ID              0
Date                  0
Status                0
Fulfilment            0
Sales Channel         0
ship-service-level    0
Category              0
Size                  0
Courier Status        0
Qty                   0
currency              0
Amount                0
ship-city             0
ship-state            0
ship-postal-code      0
ship-country          0
B2B                   0
fulfilled-by          0
dtype: int64
```

1. Sales Overview:To Understand the overall sales performance, trends, and patterns over time.

In [21]:
```
df['Date'] = pd.to_datetime(df['Date'], format='%m-%d-%y',errors="coerce")
```

In [22]:
```
df = df.dropna(subset=['Date'])
```

In [23]:
```
sales_by_date = df.groupby('Date').agg({
    'Order ID': 'count',
    'Amount': 'sum'
}).rename(columns={'Order ID': 'Order Count', 'Amount': 'Total Sales Amount'})
fig, ax1 = plt.subplots(figsize=(14, 7))
ax1.set_xlabel('Date')
ax1.set_ylabel('Order Count', color='tab:blue')
ax1.plot(sales_by_date.index, sales_by_date['Order Count'], color='tab:blue', label='Order
ax1.tick_params(axis='y', labelcolor='tab:blue')
ax2 = ax1.twinx()
ax2.set_ylabel('Total Sales Amount (INR)', color='tab:orange')
ax2.plot(sales_by_date.index, sales_by_date['Total Sales Amount'], color='tab:orange', labe
ax2.tick_params(axis='y', labelcolor='tab:orange')
fig.tight_layout()
plt.title('Sales Trends Over Time')
plt.show()
```



This shows Peak Sales in the month of April.

In [24]:
```
df1['Date'] = pd.to_datetime(df1['Date'], errors='coerce')
print(df1['Date'].isna().sum())
```

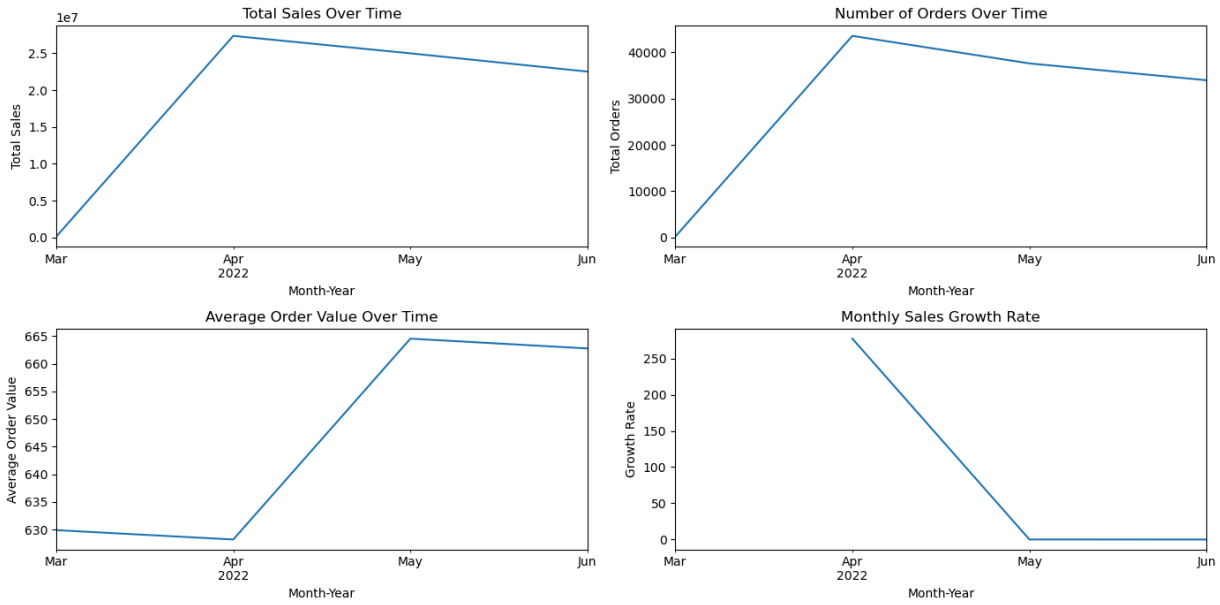
```
0

C:\Users\Dell\AppData\Local\Temp\ipykernel_31296\2111675325.py:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df1['Date'] = pd.to_datetime(df1['Date'], errors='coerce')
```

In [25]:
```
df1['Year'] = df1['Date'].dt.year
df1['Month'] = df1['Date'].dt.month
df1['Day'] = df1['Date'].dt.day
df1['Week'] = df1['Date'].dt.isocalendar().week
df1['Month-Year'] = df1['Date'].dt.to_period('M')
```

In [26]:

```
print(df1['Date'].dtype)

df1['Date'] = pd.to_datetime(df1['Date'], errors='coerce')

num_nat = df1['Date'].isna().sum()
print(f"Number of NaT values: {num_nat}")

df1.dropna(subset=['Date'], inplace=True)

df1['Year'] = df1['Date'].dt.year
df1['Month'] = df1['Date'].dt.month
df1['Day'] = df1['Date'].dt.day
df1['Week'] = df1['Date'].dt.isocalendar().week
df1['Month-Year'] = df1['Date'].dt.to_period('M')

monthly_sales = df1.groupby('Month-Year').agg({'Amount': 'sum', 'Order ID': 'count'}).renam
monthly_sales['Average Order Value'] = monthly_sales['Total Sales'] / monthly_sales['Total 

monthly_sales['Monthly Sales Growth'] = monthly_sales['Total Sales'].pct_change()

plt.figure(figsize=(14, 7))

# Total Sales Over Time
plt.subplot(2, 2, 1)
monthly_sales['Total Sales'].plot()
plt.title('Total Sales Over Time')
plt.xlabel('Month-Year')
plt.ylabel('Total Sales')

# Number of Orders Over Time
plt.subplot(2, 2, 2)
monthly_sales['Total Orders'].plot()
plt.title('Number of Orders Over Time')
plt.xlabel('Month-Year')
plt.ylabel('Total Orders')

# Average Order Value Over Time
plt.subplot(2, 2, 3)
monthly_sales['Average Order Value'].plot()
plt.title('Average Order Value Over Time')
plt.xlabel('Month-Year')
plt.ylabel('Average Order Value')

# Sales Growth Rate Over Time
plt.subplot(2, 2, 4)
monthly_sales['Monthly Sales Growth'].plot()
plt.title('Monthly Sales Growth Rate')
plt.xlabel('Month-Year')
plt.ylabel('Growth Rate')

plt.tight_layout()
plt.show()

# Calculate specific insights
peak_sales_month = monthly_sales['Total Sales'].idxmax()
peak_sales_value = monthly_sales['Total Sales'].max()

average_sales_per_month = monthly_sales['Total Sales'].mean()
average_orders_per_month = monthly_sales['Total Orders'].mean()

print(f"Peak sales month: {peak_sales_month}")
print(f"Peak sales value: {peak_sales_value:.2f}")
print(f"Average sales per month: {average_sales_per_month:.2f}")
print(f"Average number of orders per month: {average_orders_per_month:.2f}")

high_growth_months = monthly_sales[monthly_sales['Monthly Sales Growth'] > 0.20]
print("Months with more than 20% sales growth:")
print(high_growth_months[['Total Sales', 'Monthly Sales Growth']])
```

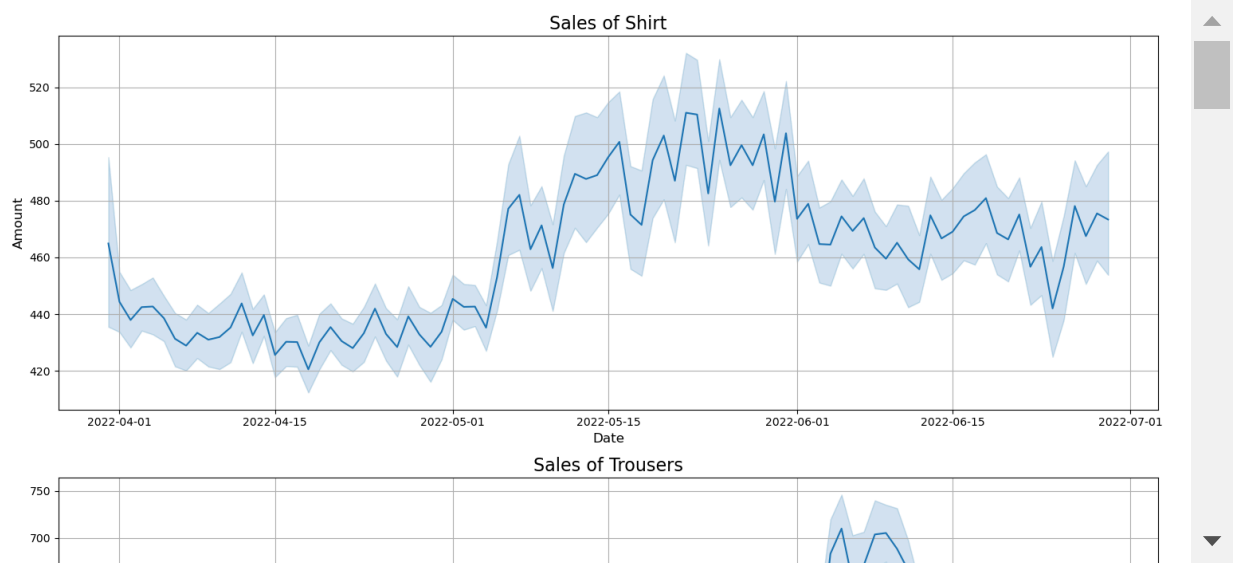
```
datetime64[ns]
Number of NaT values: 0
```




```
Peak sales month: 2022-04
Peak sales value: 27371152.00
Average sales per month: 18746710.50
Average number of orders per month: 28830.75
Months with more than 20% sales growth:
            Total Sales  Monthly Sales Growth
Month-Year
2022-04      27371152.0            277.555602
```

In [27]:

```
fig, ax = plt.subplots(9, 1, figsize=(15, 50))

cat = df1['Category'].unique().tolist()
cat = iter(cat)

for i in range(9):
    a = next(cat)
    df = df1[df1['Category'] == a].copy()
    sns.lineplot(x='Date', y='Amount', data=df, ax=ax[i])
    ax[i].set_title(f'Sales of {a}', fontsize=16)
    ax[i].set_xlabel('Date', fontsize=12)
    ax[i].set_ylabel('Amount', fontsize=12)
    ax[i].grid(True)

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

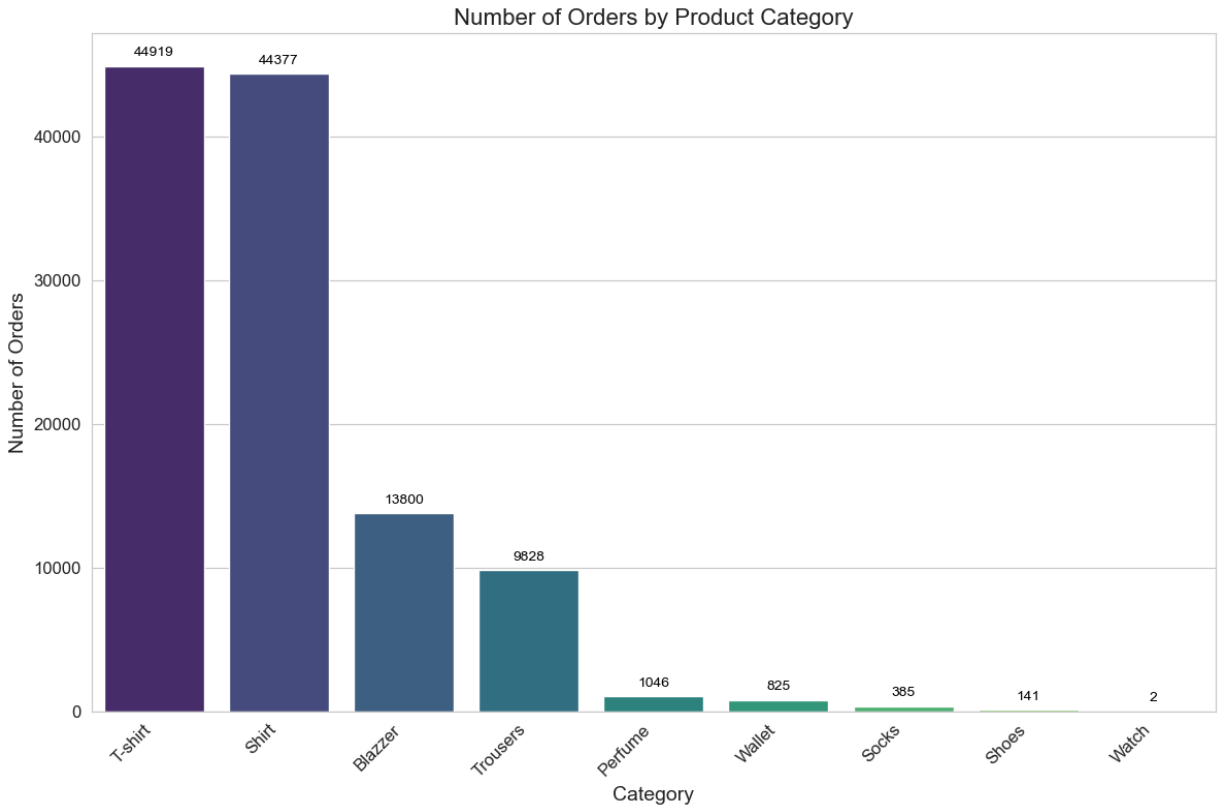



```
. The average sales of each category of product is good in between 1-05-2022 to 1-06-2022.
```

```
. The watch has no sales as it is the least bought product.
```

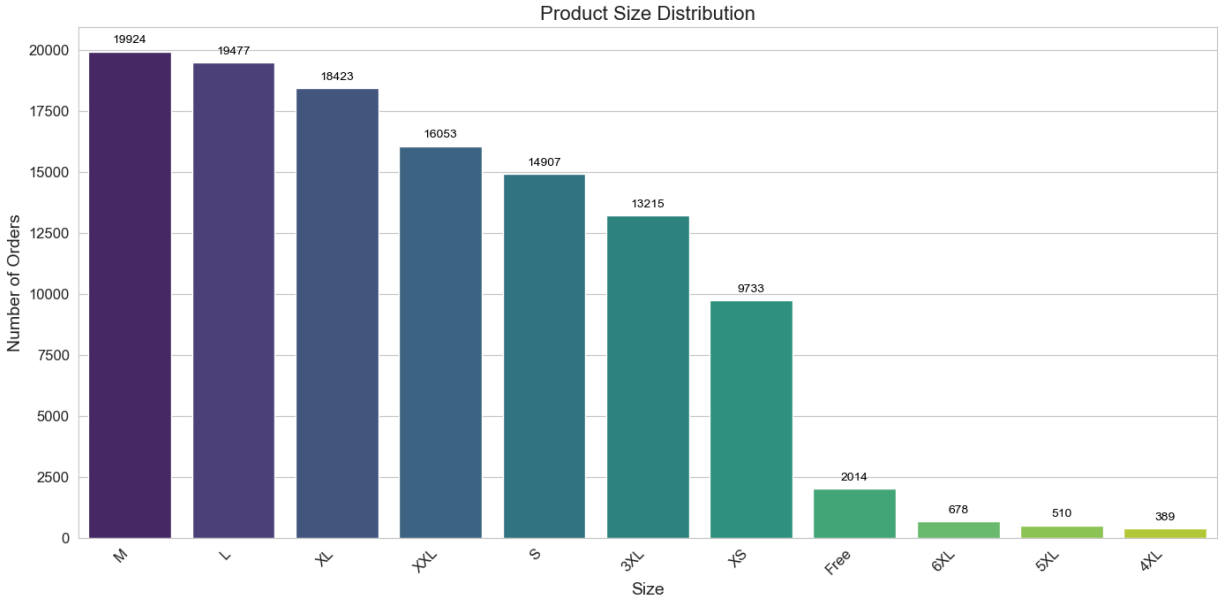
2. Product Analysis: Analyze the distribution of product categories, sizes, and quantities sold to identify popular products.

In [28]:
```
category_orders = df1['Category'].value_counts()
plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")

ax = sns.barplot(x=category_orders.index, y=category_orders.values, palette="viridis")
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',
                fontsize=10, color='black')
plt.title('Number of Orders by Product Category', fontsize=16)
plt.xlabel('Category', fontsize=14)
plt.ylabel('Number of Orders', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



This graph shows the number of orders for different categories, where Tshirt has the maximum number of orders and Watch has the minimum number of orders.

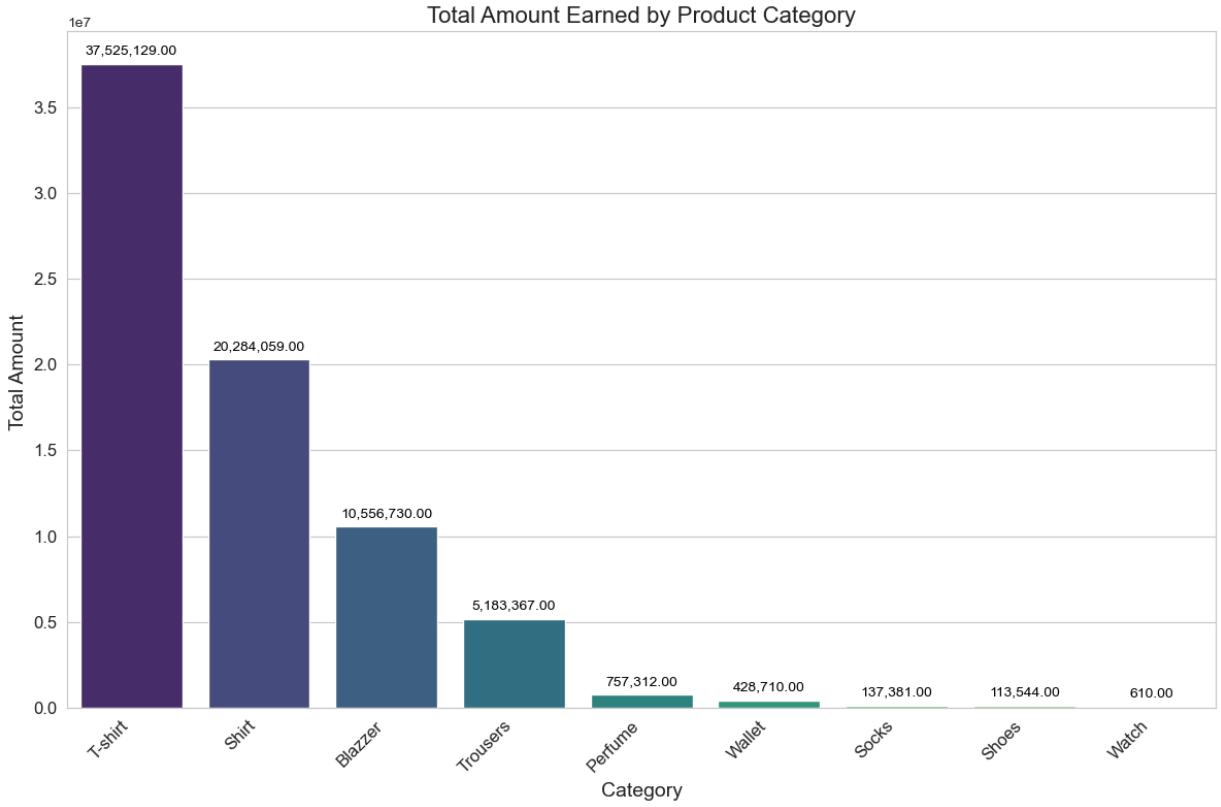
In [29]:
```
size_distribution = df1['Size'].value_counts()
plt.figure(figsize=(14, 7))
sns.set_style("whitegrid")
ax = sns.barplot(x=size_distribution.index, y=size_distribution.values, palette="viridis")
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9),
                textcoords='offset points',
                fontsize=10, color='black')

plt.title('Product Size Distribution', fontsize=16)
plt.xlabel('Size', fontsize=14)
plt.ylabel('Number of Orders', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



This graph shows the distribution of product size, where M has maximum number of orders and 4XL has minimum.

In [30]:

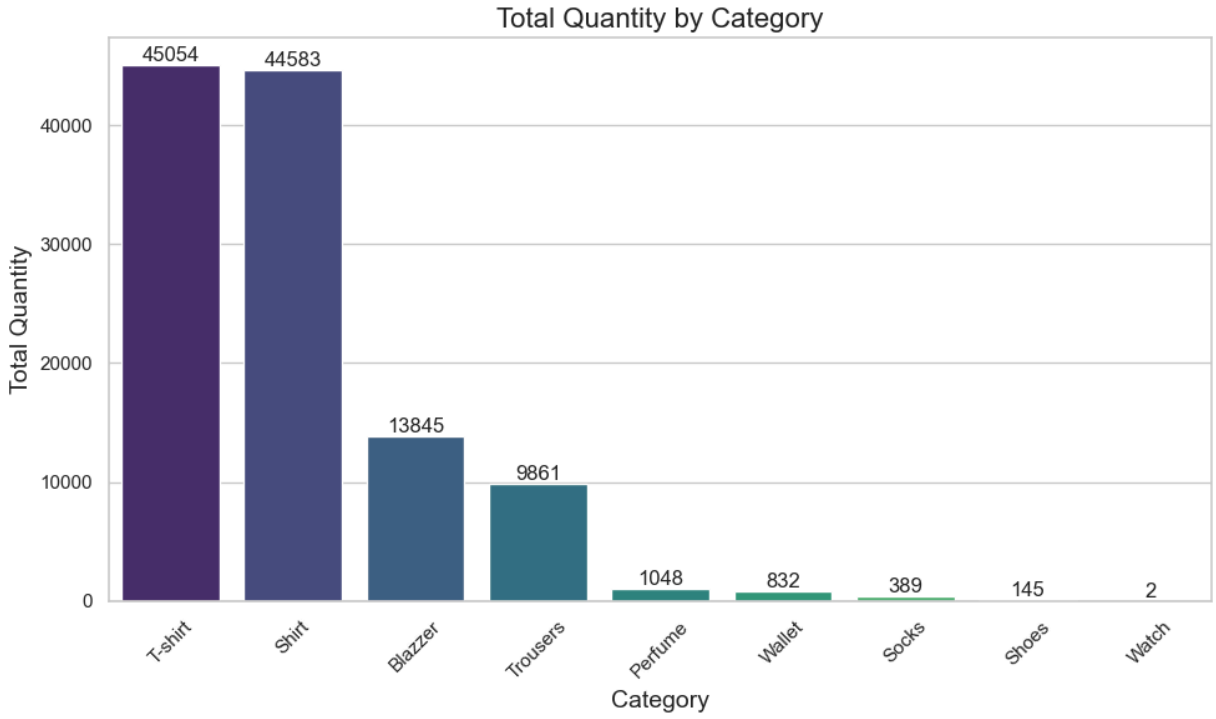
```
category_amount = df1.groupby("Category")["Amount"].sum().sort_values(ascending=False)


plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")
ax = sns.barplot(x=category_amount.index, y=category_amount.values, palette="viridis")
for p in ax.patches:
    ax.annotate("{:,.2f}".format(p.get_height()),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 9),
                textcoords='offset points',
                fontsize=10, color='black')

plt.title('Total Amount Earned by Product Category', fontsize=16)
plt.xlabel('Category', fontsize=14)
plt.ylabel('Total Amount ', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



This shows the total amount earned by different categories where T-shirt has maximum Sales Amount.

In [54]:
```
category_amount = df1.groupby("Category")["Qty"].sum().reset_index()
category_amount_sorted = category_amount.sort_values(by='Qty', ascending=False)
print(category_amount_sorted)
sns.set_style("whitegrid")
plt.figure(figsize=(10, 6))
bar_plot = sns.barplot(x="Category", y="Qty", data=category_amount_sorted, palette="viridis
bar_plot.set_title('Total Quantity by Category', fontsize=16)
bar_plot.set_xlabel('Category', fontsize=14)
bar_plot.set_ylabel('Total Quantity', fontsize=14)
plt.xticks(rotation=45)
for container in bar_plot.containers:
    bar_plot.bar_label(container)
plt.tight_layout()
plt.show()
```

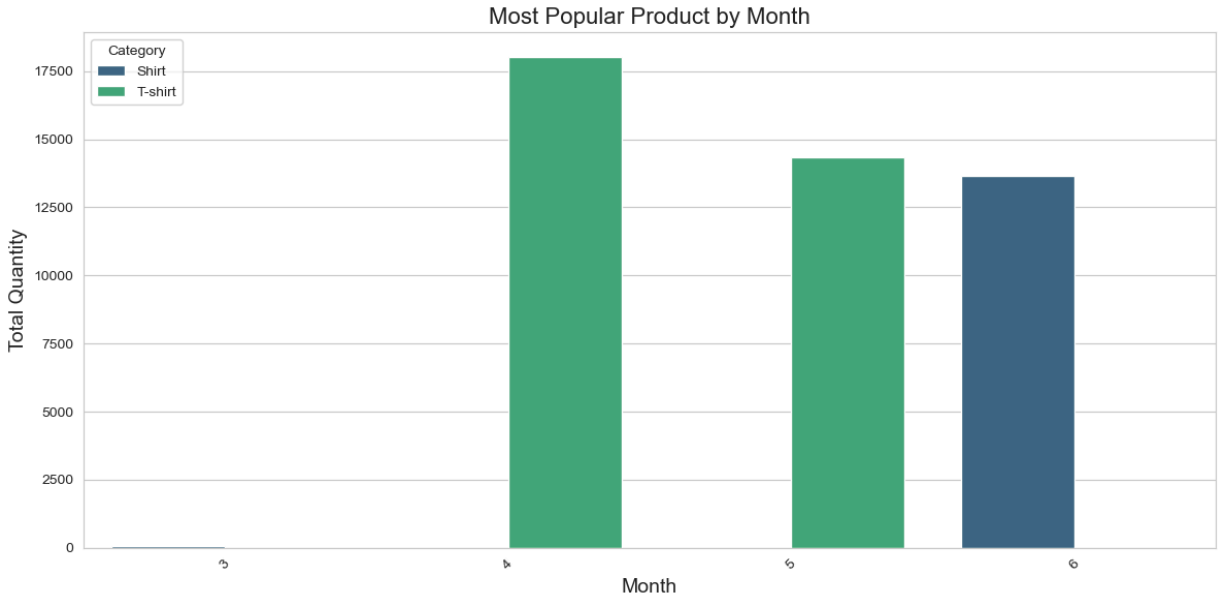
```
   Category    Qty
5   T-shirt  45054
2     Shirt  44583
0   Blazzer  13845
6  Trousers   9861
1   Perfume   1048
7    Wallet    832
4     Socks    389
3     Shoes    145
8     Watch      2
```



This graph shows T-shirt has maximum quantity So we can focus on the Marketing strategies and Product Diversification for Tshirt

In [32]:
```
monthly_category_amount = df1.groupby(['Month', 'Category'])['Qty'].sum().reset_index()
most_popular_by_month = monthly_category_amount.loc[monthly_category_amount.groupby('Month'
print(most_popular_by_month)
sns.set_style("whitegrid")
plt.figure(figsize=(12, 6))
bar_plot = sns.barplot(x="Month", y="Qty", hue="Category", data=most_popular_by_month, pale
bar_plot.set_title('Most Popular Product by Month', fontsize=16)
bar_plot.set_xlabel('Month', fontsize=14)
bar_plot.set_ylabel('Total Quantity', fontsize=14)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
    Month Category    Qty
2       3    Shirt     71
11      4  T-shirt  18015
19      5  T-shirt  14354
24      6    Shirt  13665
```



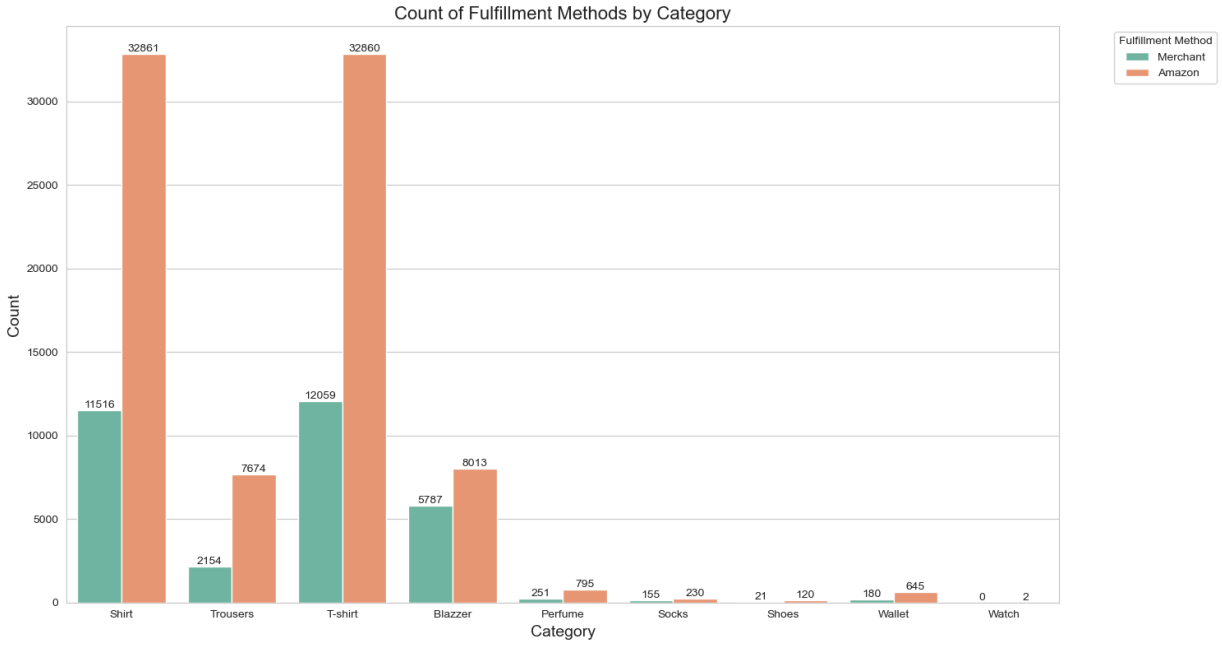
This shows Famous Product by month Where T-shirt has maximum sales in April and May but in June the Sales of Shirt is Maximum

3. Fulfillment Analysis: Investigate the fulfillment methods used and their effectiveness in delivering orders.

In [33]:
```
df1.groupby("Category")["Fulfilment"].value_counts()
```

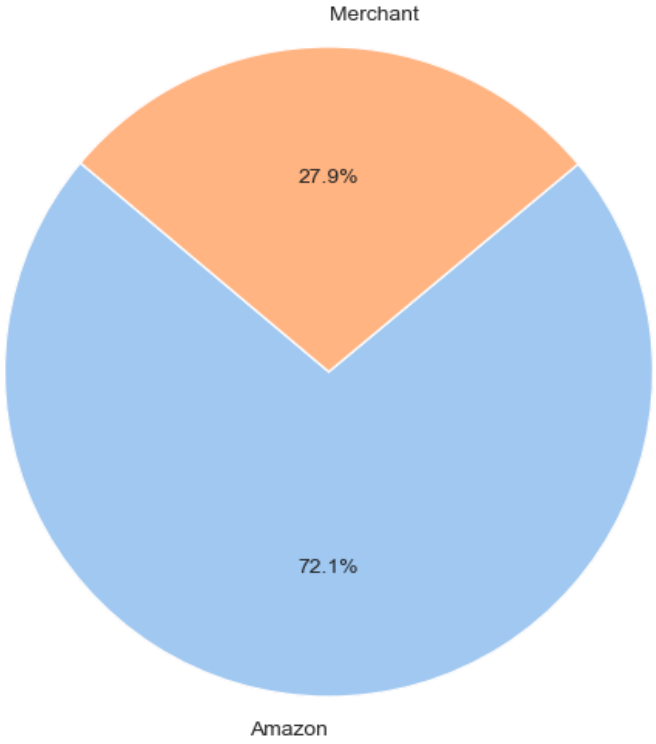
Out[33]:
```
Category  Fulfilment
Blazzer   Amazon         8013
          Merchant       5787
Perfume   Amazon          795
          Merchant        251
Shirt     Amazon        32861
          Merchant      11516
Shoes     Amazon          120
          Merchant         21
Socks     Amazon          230
          Merchant        155
T-shirt   Amazon        32860
          Merchant      12059
Trousers  Amazon         7674
          Merchant       2154
Wallet    Amazon          645
          Merchant        180
Watch     Amazon            2
Name: count, dtype: int64
```

In [34]:
```
sns.set_style("whitegrid")
palette = sns.color_palette("Set2")
plt.figure(figsize=(15, 8))
ax = sns.countplot(data=df1, x="Category", hue="Fulfilment", palette=palette)
for container in ax.containers:
    ax.bar_label(container)
plt.title('Count of Fulfillment Methods by Category', fontsize=16)
plt.xlabel('Category', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.legend(title='Fulfillment Method', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



This bar chart shows the distribution of fulfilment methods used for Products

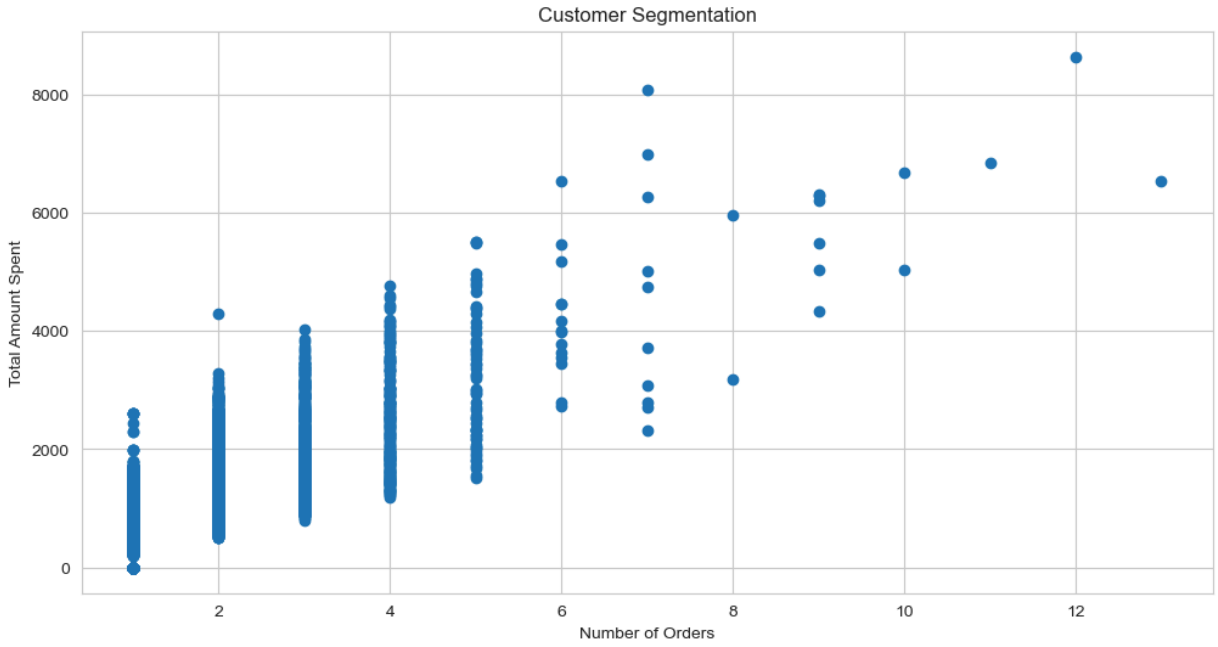
In [35]:
```
fulfillment_counts = df1['Fulfilment'].value_counts(normalize=True) * 100
sns.set_style("whitegrid")
colors = sns.color_palette("pastel")
plt.figure(figsize=(10, 6))
plt.pie(fulfillment_counts, labels=fulfillment_counts.index, autopct='%1.1f%%', startangle=
plt.axis('equal')
plt.show()
```



Fulfilment by Amazon is the most efficient method.

4. Customer Segmentation: Segment customers based on their buying behaviour, location, and other relevant factors.

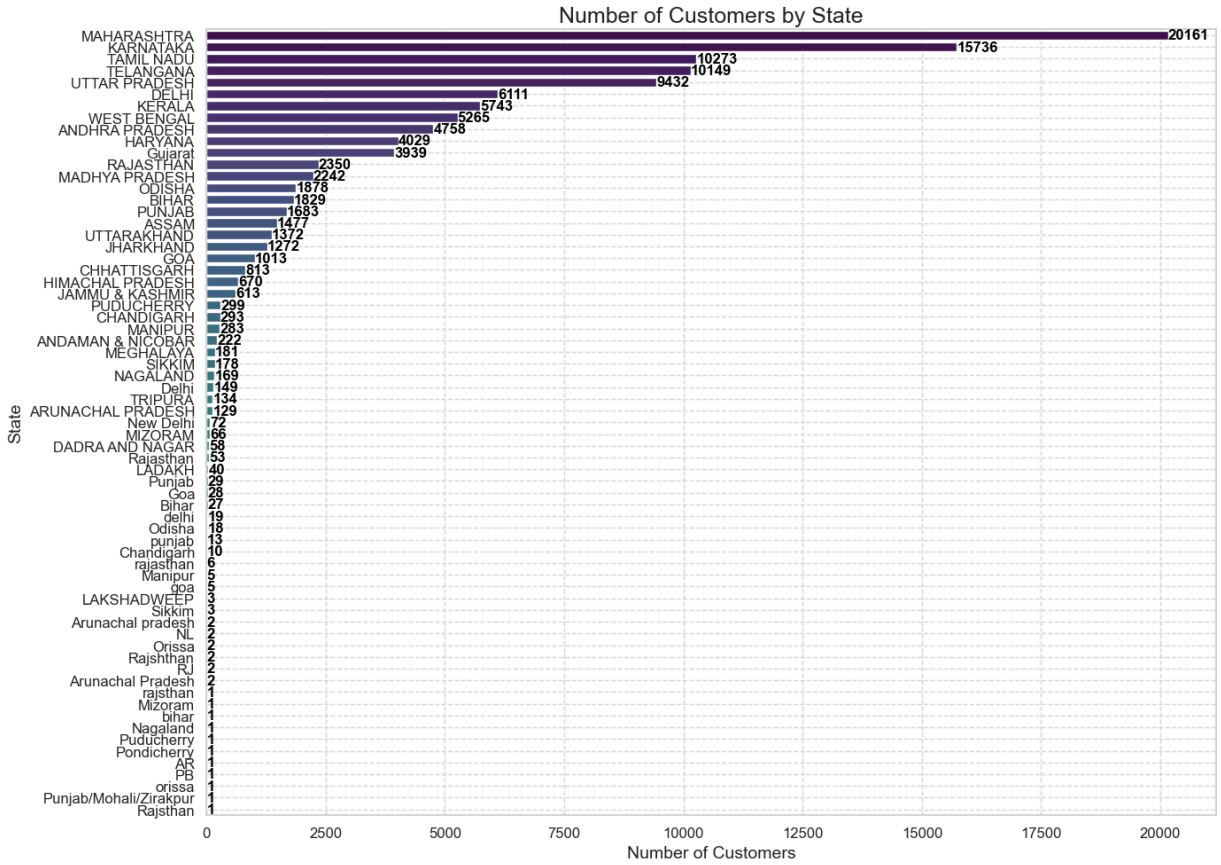
In [39]:
```
customer_segments = df1.groupby('Order ID').agg({
    'Amount': 'sum',
    'Qty': 'sum'
}).rename(columns={'Amount': 'total_spent', 'Qty': 'order_count'})

# Visualize customer segmentation
plt.figure(figsize=(12, 6))
plt.scatter(customer_segments['order_count'], customer_segments['total_spent'])
plt.title('Customer Segmentation')
plt.xlabel('Number of Orders')
plt.ylabel('Total Amount Spent')
plt.grid(True)
plt.show()
```




```
This scatter plot illustrates customer segmentation based on the number of orders and the total amount spent.amount they spent.
Positive Correlation: There's a clear trend showing that as the number of orders increases, the total amount spent also increases. This suggests that frequent buyers tend to spend more overall.
Customer Distribution: Most customers have placed between 1 to 6 orders, with spending generally ranging from ₹1,000 to ₹4,000. Outliers are seen where customers with a higher number of orders have spent significantly more, some exceeding ₹8,000.
Insight: The data indicates a potential opportunity to target customers with fewer orders through personalized marketing to encourage repeat purchases, thereby increasing their total spend.
```

In [40]:

```
state_customer_count = df1['ship-state'].value_counts().reset_index()
state_customer_count.columns = ['Ship State', 'Customer Count']
plt.figure(figsize=(14, 10))
sns.set(style="whitegrid")
barplot = sns.barplot(x='Customer Count', y='Ship State', data=state_customer_count, palett
plt.title('Number of Customers by State', fontsize=18)
plt.xlabel('Number of Customers', fontsize=14)
plt.ylabel('State', fontsize=14)
for index, value in enumerate(state_customer_count['Customer Count']):
    barplot.text(value + 0.1, index, str(value), va='center', ha='left', color='black', fon
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```

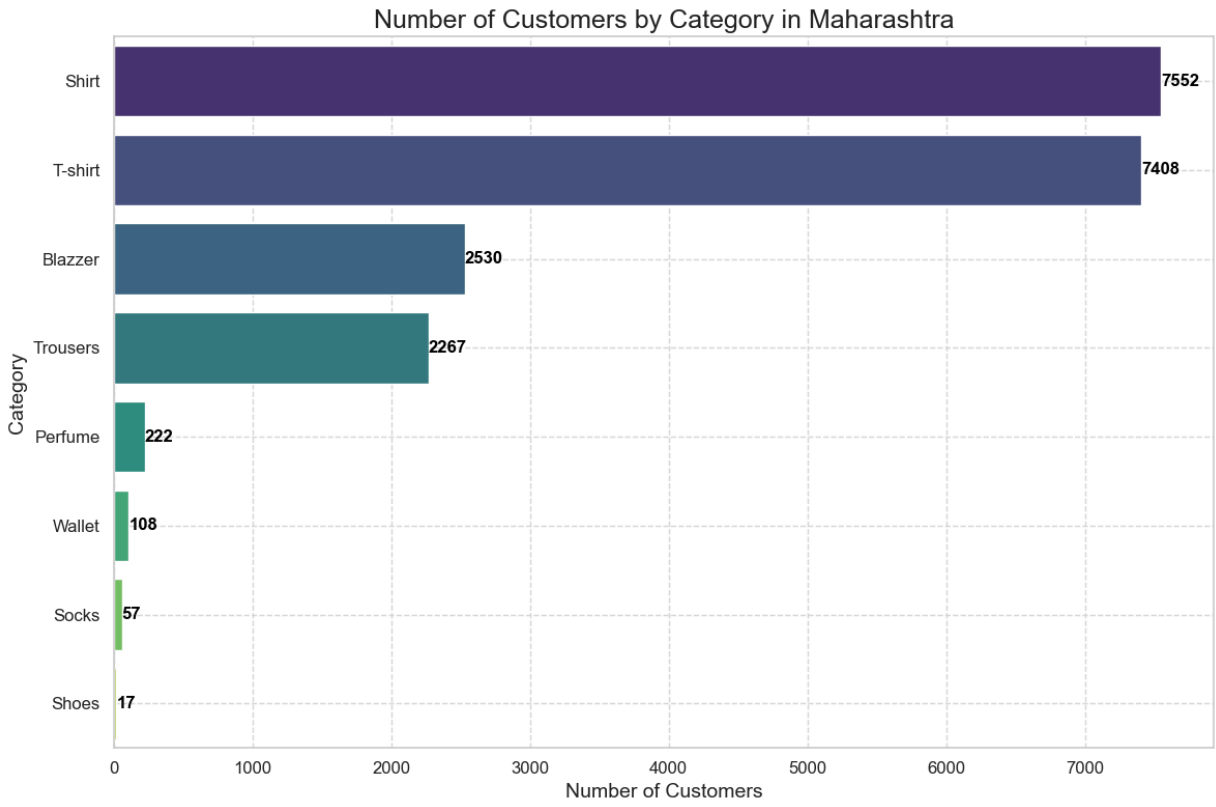



```
Top States:
Maharashtra leads with the highest number of customers (20,161).
Karnataka follows with 15,736 customers.
Tamil Nadu and Telangana also have significant customer bases with 10,273 and 10,149
customers.
Mid-Tier States:
Uttar Pradesh, Kerala, and West Bengal have moderate customer numbers ranging from
approximately 5,000 to 9,000.
Lower-Tier States:
Several states, including Haryana, Gujarat, Rajasthan, and Madhya Pradesh, have fewer
than 5,000 customers.
States like Sikkim, Nagaland, and Tripura have customer numbers below 200.
Least Populated States:
Lakshadweep, Sikkim, and Mizoram have the smallest customer bases, each with fewer than
100 customers.
Recommendations:
Focus on High Customer States: Develop targeted marketing strategies and promotions in
states like Maharashtra, Karnataka, Tamil Nadu, and Telangana to further increase sales
and customer engagement.
Expand in Mid-Tier States: Strengthen presence and explore growth opportunities in states
with moderate customer bases such as Uttar Pradesh, Kerala, and West Bengal.
```

```
Address Low Customer States: Investigate the low customer numbers in states with minimal representation to identify potential barriers and tailor specific strategies to attract new customers.
```

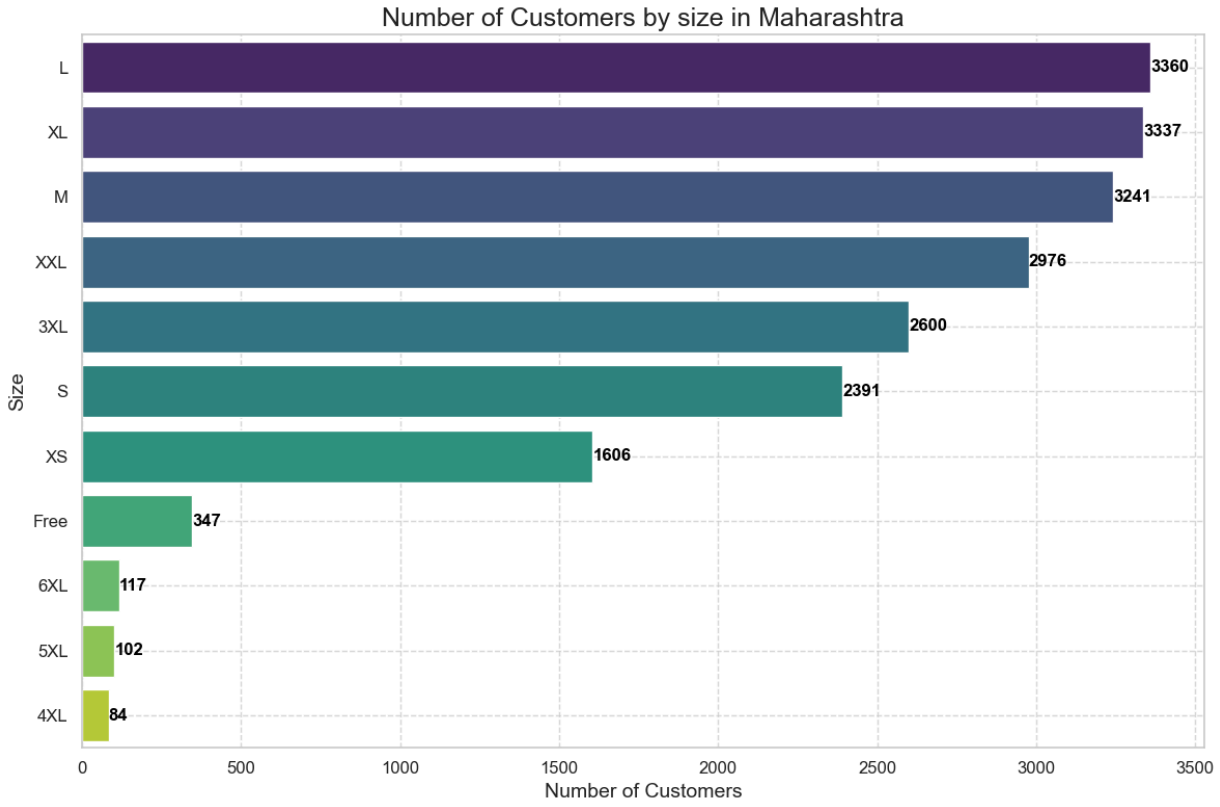
In [41]:

```
maharashtra_orders = df1[df1['ship-state'] == 'MAHARASHTRA']
category_customer_count = maharashtra_orders['Category'].value_counts().reset_index()
category_customer_count.columns = ['Category', 'Customer Count']
plt.figure(figsize=(12, 8))
sns.set(style="whitegrid")
barplot = sns.barplot(x='Customer Count', y='Category', data=category_customer_count, palet
plt.title('Number of Customers by Category in Maharashtra', fontsize=18)
plt.xlabel('Number of Customers', fontsize=14)
plt.ylabel('Category', fontsize=14)
for index, value in enumerate(category_customer_count['Customer Count']):
    barplot.text(value + 0.1, index, str(value), va='center', ha='left', color='black', fon
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



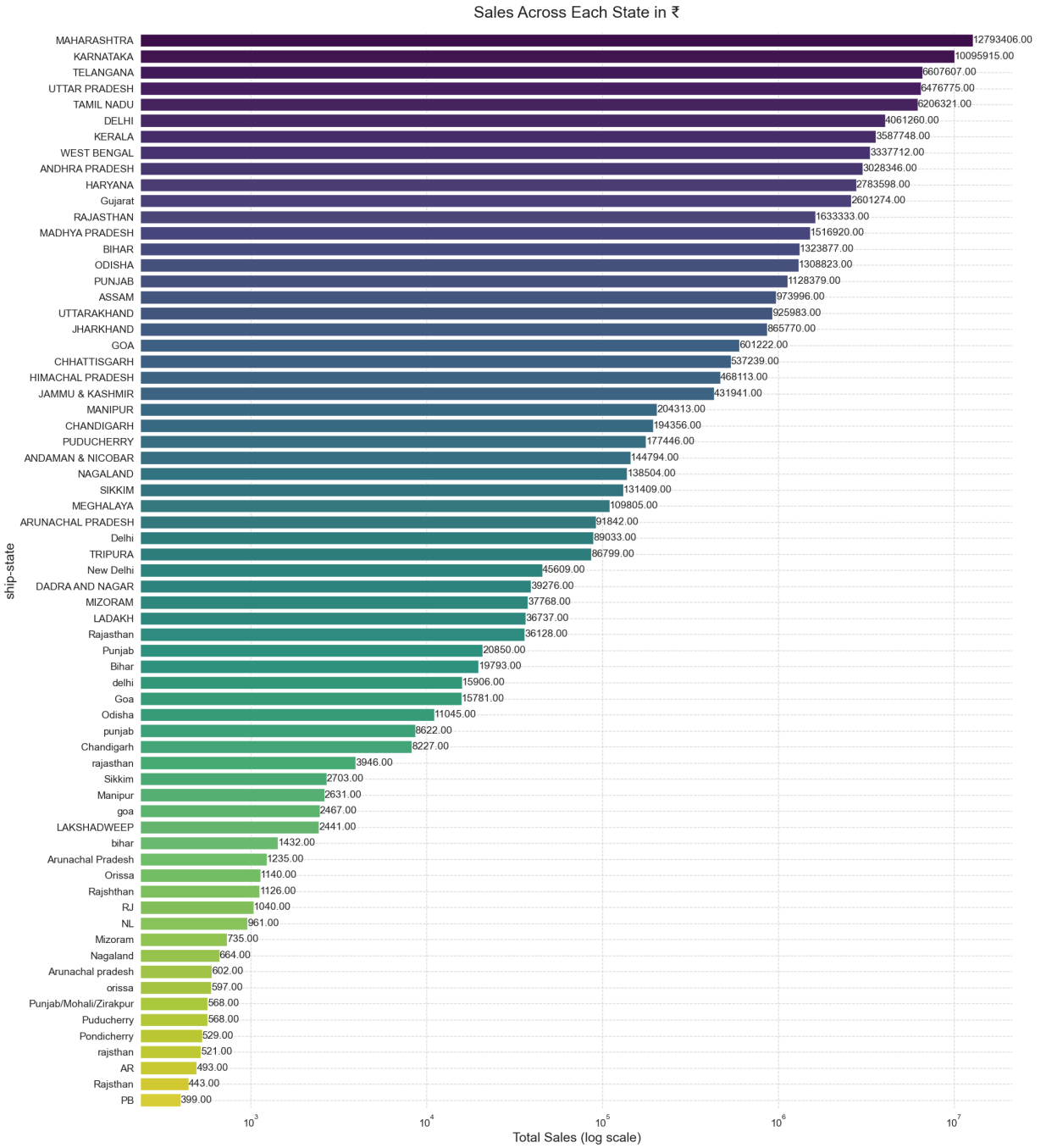

```
To focus on the state Maharashtra with maximum sales and orders, here the Shirt has maximum number of customers.
```

In [42]:
```
maharashtra_orders = df1[df1['ship-state'] == 'MAHARASHTRA']
category_customer_count = maharashtra_orders['Size'].value_counts().reset_index()
category_customer_count.columns = ['Size', 'Customer Count']
plt.figure(figsize=(12, 8))
sns.set(style="whitegrid")
barplot = sns.barplot(x='Customer Count', y='Size', data=category_customer_count, palette='
plt.title('Number of Customers by size in Maharashtra', fontsize=18)
plt.xlabel('Number of Customers', fontsize=14)
plt.ylabel('Size', fontsize=14)
for index, value in enumerate(category_customer_count['Customer Count']):
    barplot.text(value + 0.1, index, str(value), va='center', ha='left', color='black', fon
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```




```
This shows in Maharashtra L size has maximum number of customers.
```

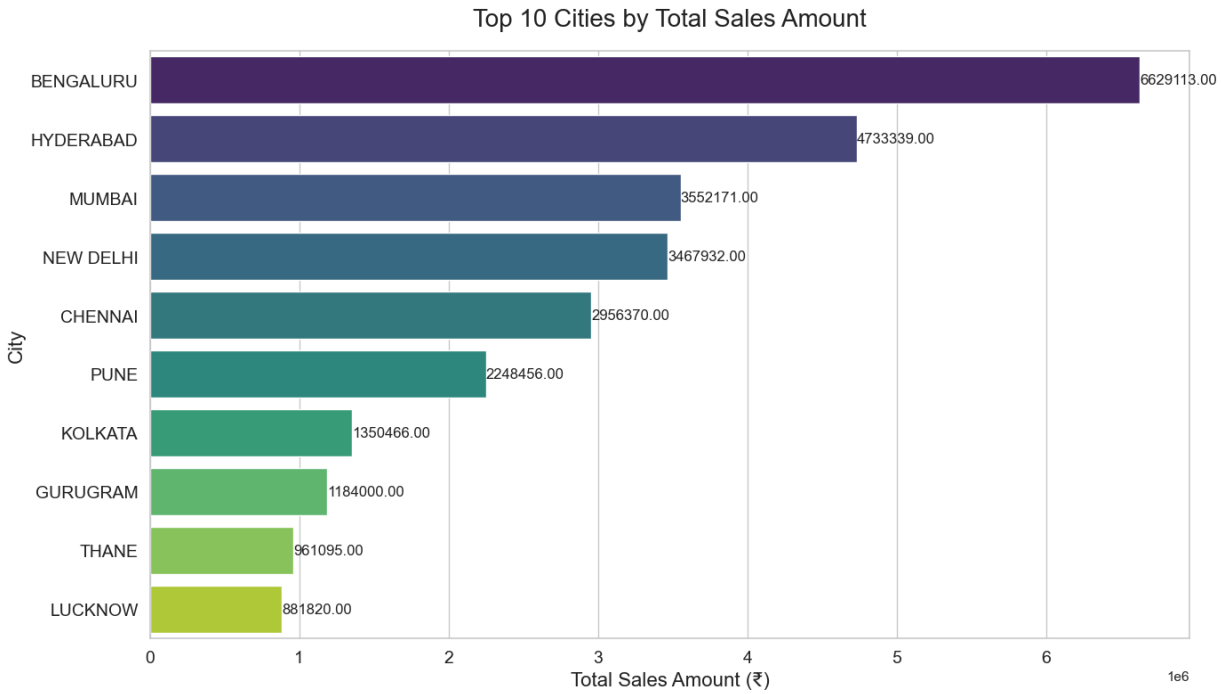
In [43]:
```
plt.figure(figsize=(20, 22))
df2 = pd.DataFrame(df1.groupby("ship-state")["Amount"].sum())
df2 = df2.sort_values(by="Amount", ascending=False)
sns.set(style="whitegrid")
axs = sns.barplot(y=df2.index, x="Amount", data=df2, palette="viridis", log=True)
for container in axs.containers:
    axs.bar_label(container, fmt="%.2f", label_type='edge', fontsize=14)
plt.ylabel("ship-state", fontsize=18)
plt.xlabel("Total Sales (log scale)", fontsize=18)
plt.title("Sales Across Each State in ₹", fontsize=22, pad=20)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
sns.despine(left=True, bottom=True)
plt.tight_layout()
plt.show()
```



In [ ]:
```
This shows Maharashtra has maximum sales.
```

In [44]:

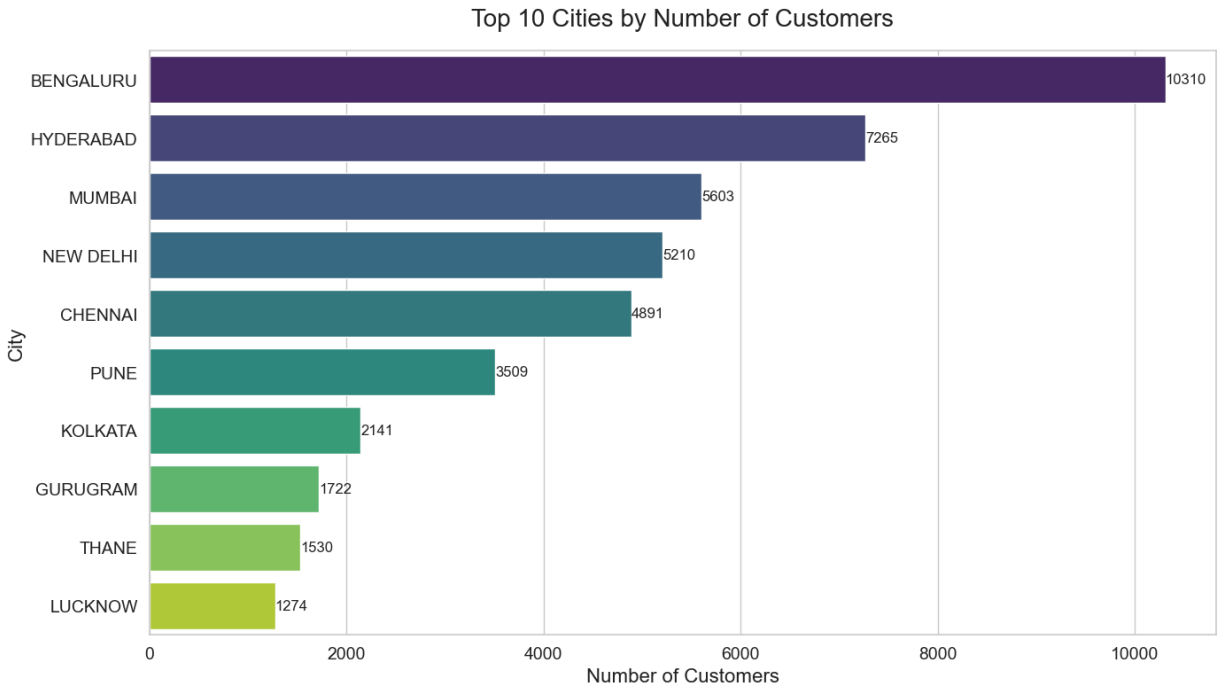
```
city_sales = df1.groupby('ship-city')['Amount'].sum().reset_index()
top_cities = city_sales.sort_values(by='Amount', ascending=False).head(10)
plt.figure(figsize=(14, 8))
sns.set(style="whitegrid")
barplot = sns.barplot(y='ship-city', x='Amount', data=top_cities, palette='viridis')
for container in barplot.containers:
    barplot.bar_label(container, fmt='%.2f', label_type='edge', fontsize=12)
plt.ylabel("City", fontsize=16)
plt.xlabel("Total Sales Amount (₹)", fontsize=16)
plt.title("Top 10 Cities by Total Sales Amount", fontsize=20, pad=20)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.tight_layout()
plt.show()
```




```
This shows the City with maximum sales amount, here Bengaluru has maximum sales amount
```

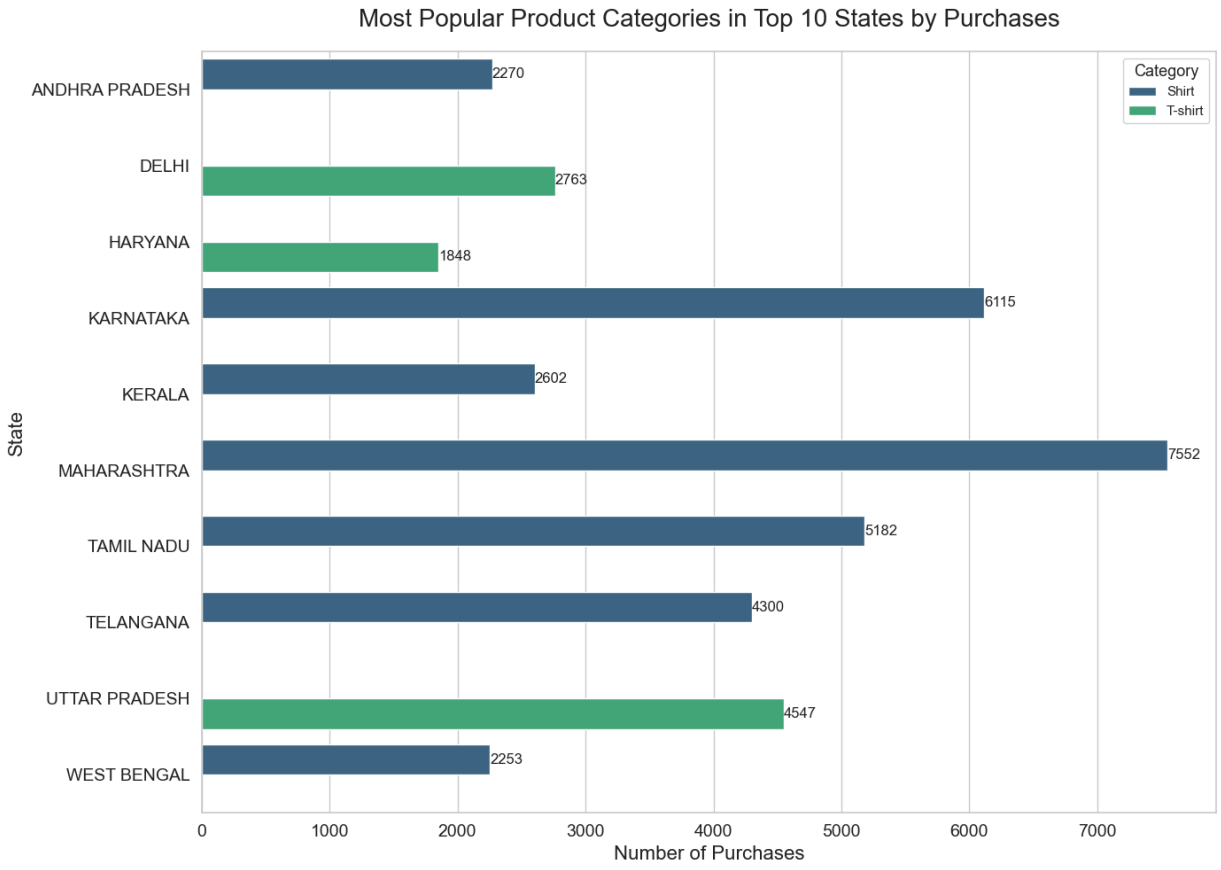
In [45]:

```
city_customers = df1.groupby('ship-city').size().reset_index(name='Customer Count')
top_cities_customers = city_customers.sort_values(by='Customer Count', ascending=False).hea
plt.figure(figsize=(14, 8))
sns.set(style="whitegrid")
barplot = sns.barplot(y='ship-city', x='Customer Count', data=top_cities_customers, palette
for container in barplot.containers:
    barplot.bar_label(container, fmt='%d', label_type='edge', fontsize=12)
plt.ylabel("City", fontsize=16)
plt.xlabel("Number of Customers", fontsize=16)
plt.title("Top 10 Cities by Number of Customers", fontsize=20, pad=20)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.tight_layout()
plt.show()
```



In [46]:

```
state_purchase_counts = df1.groupby('ship-state').size().reset_index(name='Total Purchases'
top_10_states = state_purchase_counts.sort_values(by='Total Purchases', ascending=False).hea
top_10_states_df = df1[df1['ship-state'].isin(top_10_states['ship-state'])]
popular_categories = top_10_states_df.groupby(['ship-state', 'Category']).size().reset_inde
most_popular_categories = popular_categories.loc[popular_categories.groupby('ship-state')['
plt.figure(figsize=(14, 10))
sns.set(style="whitegrid")
barplot = sns.barplot(y='ship-state', x='Counts', hue='Category', data=most_popular_categor
for container in barplot.containers:
    barplot.bar_label(container, fmt='%d', label_type='edge', fontsize=12)
plt.ylabel("State", fontsize=16)
plt.xlabel("Number of Purchases", fontsize=16)
plt.title("Most Popular Product Categories in Top 10 States by Purchases", fontsize=20, pad
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.legend(title='Category', title_fontsize='13', fontsize='11', loc='upper right')
plt.tight_layout()
plt.show()
```




```
Maharashtra:
The state has the highest number of purchases in the Shirt category with 7,552 purchases.
Karnataka:
Follows Maharashtra with 6,115 purchases in the Shirt category.
Delhi:
Notably, T-shirts are more popular with 2,763 purchases compared to Shirts.
Uttar Pradesh:
Similar to Delhi, T-shirts lead with 4,547 purchases.
Tamil Nadu and Telangana:
These states have significant purchases in the Shirt category with 5,182 and 4,300.
Kerala, Andhra Pradesh, West Bengal, and Haryana:
All these states show a higher preference for Shirts over T-shirts.
Actionable Recommendations:
Maharashtra and Karnataka: Focus on maintaining the inventory and introducing new styles
in the Shirt category to sustain and grow the market.
Delhi and Uttar Pradesh: Increase marketing efforts and promotions for T-shirts given
their popularity in these states.
```

```
Other States: Tailor marketing strategies to emphasize the popular product category in
each state, potentially boosting sales by aligning with local preferences.
```

6. Business Insights: Provide actionable insights and recommendations based on the analysis to optimize sales strategies, improve customer satisfaction, and enhance overall business performance.

1. Peak sales periods are observed in April. Plan inventory and marketing campaigns accordingly.

2. Product categories T-shirt, Shirt, Blazzer are most popular. Consider expanding these categories.

3. High-value customers are those with more than 2 orders and total spending above 828.00. Focus on loyalty programs for these customers.

4. States MAHARASHTRA, KARNATAKA, TELANGANA and cities BENGALURU, HYDERABAD, MUMBAI show high sales. Consider increasing marketing efforts and presence in these areas.

5. Maharashtra has the maximum number of Customers with maximum Sales, the famous Product in Maharashtra is Shirt and the most borrowed size is "L"

6. The Fulfilment by Amazon is the most efficient fulfilment method.