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

import warnings

pd.set_option('display.max_rows', None)
```

Summery

Analytixlabs has been hired by one of India's leading online marketplaces to provide data-driven insights into their business performance. The client is seeking assistance in measuring, managing, and analyzing various aspects of their business, including customer, seller, product, and channel behaviors.

The available data spans from September 2016 to October 2018 and comprises several key tables:

- Customers: Contains information about the customers, likely including demographics, purchase history, and other relevant details.
- Sellers: Provides details about the sellers on the platform, such as their profiles, products offered, and performance metrics.
- Products: Includes information on the products available for sale, including attributes like category, price, and description.
- Orders: Contains data related to orders placed on the platform, including order IDs, product IDs, order status, and dates.
- Order_Items: Provides order-level information, likely including details about the items purchased within each order.
- Order_Payments: Contains information about payments made for orders, including payment methods, amounts, and dates.
- Order_Review_Ratings: Includes customer ratings and reviews at the order level, which can provide insights into customer satisfaction and product quality.
- Geo-Location: Provides location details, which could be used to analyze regional variations in customer behavior or market trends.

As an analyst for this project, the tasks would involve cleaning and processing the data as necessary before conducting analysis. The goal is to derive actionable insights to help the client make informed decisions and optimize their business operations. This could include identifying trends, understanding customer preferences, evaluating seller performance, optimizing product offerings, and enhancing the overall customer experience.

Business Objective

Business Objective:

The below are few Sample business questions to be addressed as part of this analysis. However this is not exhaustive list and you can add as many as analysis and provide insights on the same.

Perform Detailed exploratory analysis:

1. Define & calculate high level metrics like (Total Revenue, Total quantity, Total products, Total categories, Total sellers, Total locations, Total channels, Totalpayment methods etc…)
2. Understanding how many new customers acquired every month
3. Understand the retention of customers on month on month basis
4. How the revenues from existing/new customers on month on month basis
5. Understand the trends/seasonality of sales, quantity by category, location, month, week, day, time, channel, payment method etc…
6. Popular Products by month, seller, state, category.
7. Popular categories by state, month
8. List top 10 most expensive products sorted by price

Performing Customers/sellers Segmentation:

1. Divide the customers into groups based on the revenue generated
2. Divide the sellers into groups based on the revenue generated Cross-Selling (Which products are selling together) Hint: We need to find which of the top 10 combinations of products are selling together in each transaction. (combination of 2 or 3 buying together)

Payment Behaviour:

1. How customers are paying?
2. Which payment channels are used by most customers?

Customer satisfaction towards category & product:

1. Which categories (top 10) are maximum rated & minimum rated?
2. Which products (top10) are maximum rated & minimum rated?
3. Average rating by location, seller, product, category, month etc. Etc..

```
# Imporing data

customers_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/CUSTOMERS.csv')

geo_location_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/GEO_LOCATION.csv')

orders_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/ORDERS.csv')

order_items_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/ORDER_ITEMS.csv')

order_payment_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/ORDER_PAYMENTS.csv')

order_review_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/ORDER_REVIEW_RATINGS.csv')

product_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/PRODUCTS.csv')

sellers_data = pd.read_csv('/content/drive/MyDrive/DataSets/Marketing Analysis/SELLERS.csv')
```

Now will go throgh each of CSV and perform EDA for same

```
customers_data.head()
```

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	Adilabad	Andhra Pradesh
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	Adoni	Andhra Pradesh
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	Akkarampalle	Andhra Pradesh
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	Akkayapalle	Andhra Pradesh
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	Alwal	Andhra Pradesh

Next steps: Generate code with customers_data View recommended plots

```
customers_data.isnull().sum()
```

```
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
```

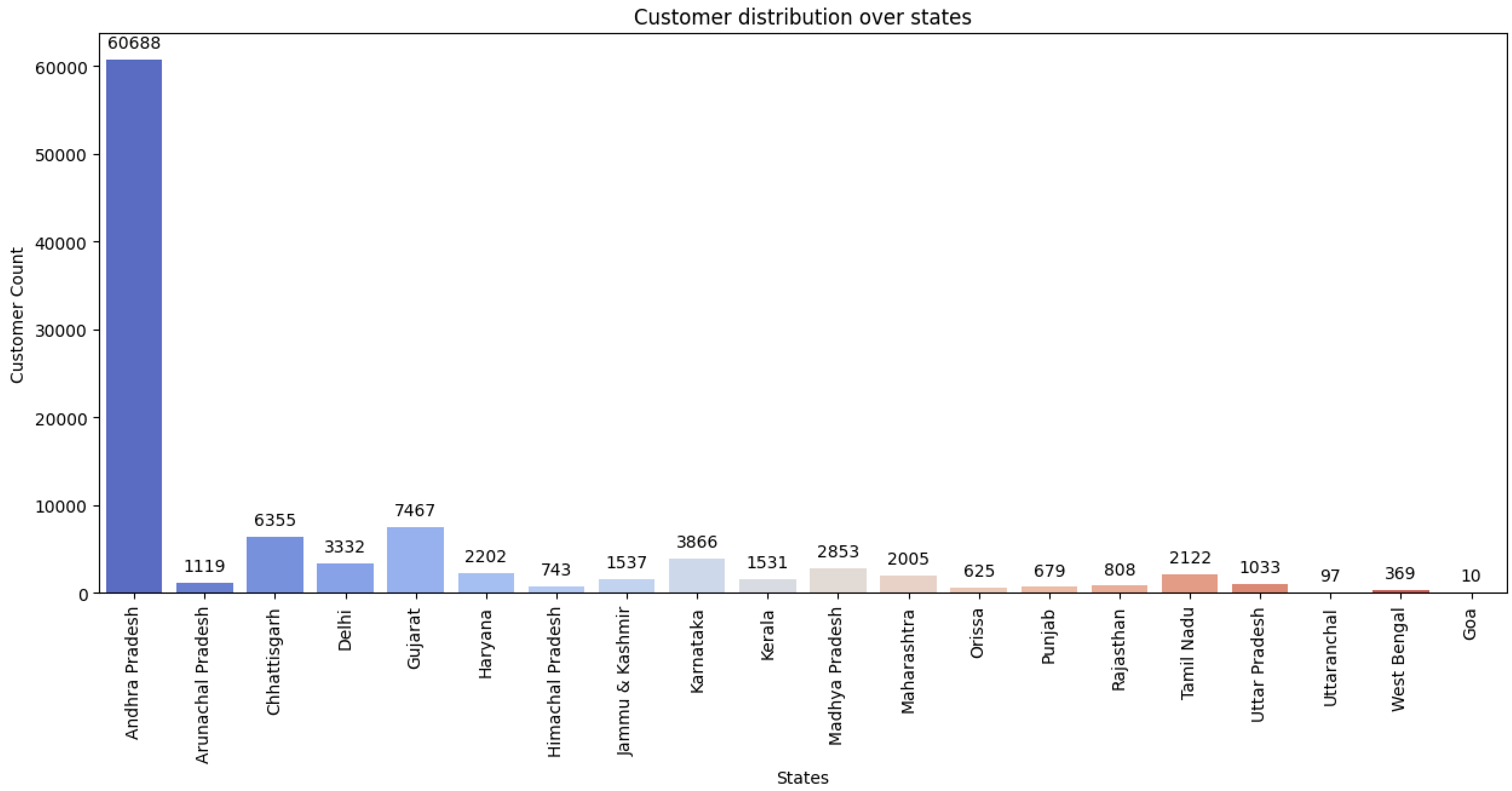
No any null values and duplicte calues present in the customers_data dataset.

EDA on customers_data

Chart 1: **Customer distribution over states**

```
states_counts = customers_data['customer_state'].value_counts()

states_counts.reset_index()

plt.figure(figsize=(15,6))
palette = "coolwarm"
ax = sns.countplot(x='customer_state',data = customers_data,palette=palette)

# Adding data labels
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, 10),
                textcoords = 'offset points')
plt.xlabel('States')
plt.ylabel('Customer Count')
plt.title('Customer distribution over states')
plt.xticks(rotation=90)

plt.show()
```

```
<ipython-input-203-7bcd45ee4686>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(x='customer_state',data = customers_data,palette=palette)
```

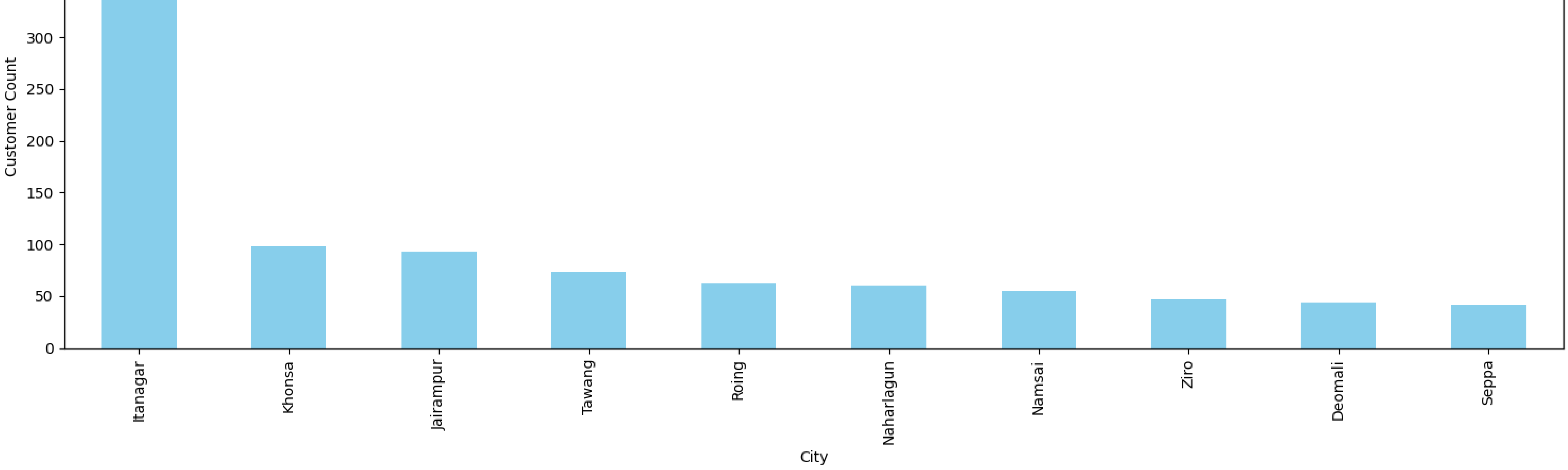


Insights from above chart:

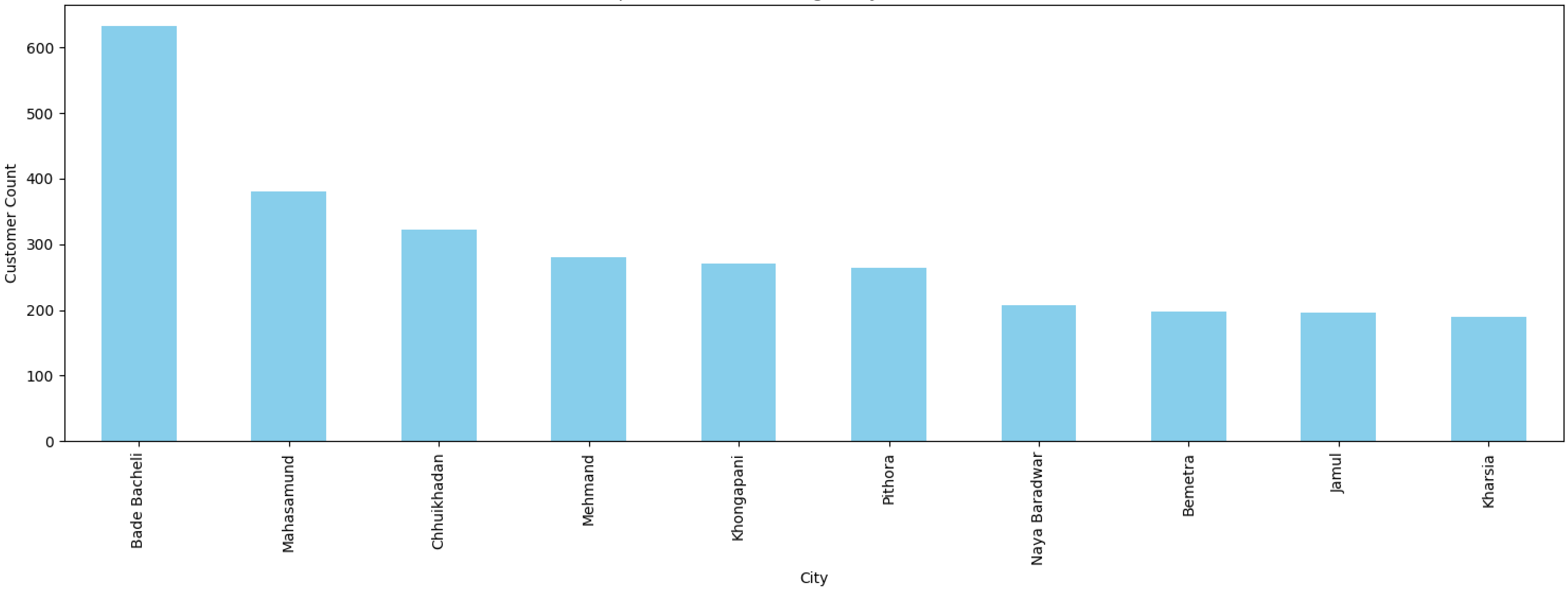
- **Andhra Pradesh** has the highest number of customers, indicating a large customer base in that region.**Gujarat, Chhattisgarh, and Karnataka** also have significant customer counts, suggesting a substantial market presence in these states.
- States with relatively lower customer counts, such as **Jammu & Kashmir, Kerala, and Arunachal Pradesh**, might present growth opportunities for expanding the customer base through targeted marketing strategies or localized offerings.**Uttaranchal and Goa** have very low customer counts, indicating potential untapped markets or areas where the platform might need to focus more attention to attract customers.

Chart 2: Customer Count Distribution Across Cities in Each State

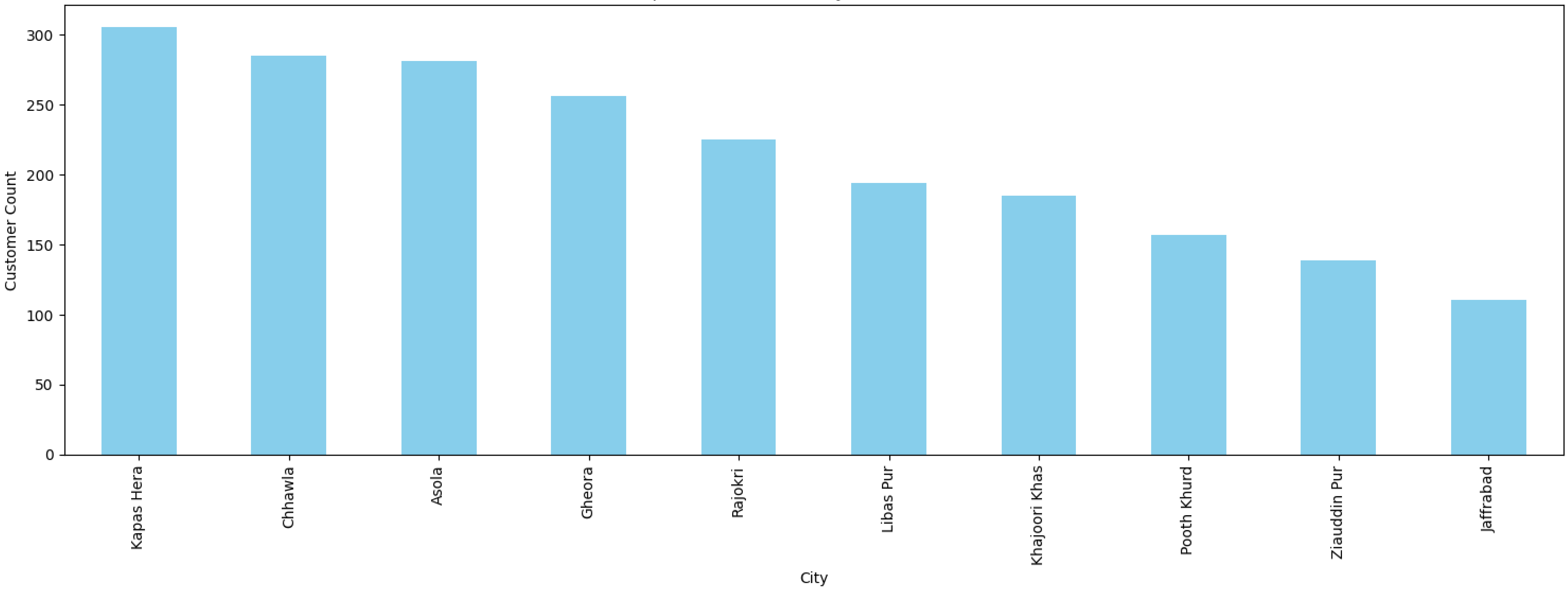
```
import matplotlib.pyplot as plt

# Get unique states from the dataset
states = customers_data['customer_state'].unique()

# Iterate over each state
for state in states:
    # Filter data for the current state
    state_data = customers_data[customers_data['customer_state'] == state]

    # Group by 'customer_city' and count number of customers in each city
    city_counts = state_data['customer_city'].value_counts().head(10)

    # Plot the bar plot
    plt.figure(figsize=(15,6))
    city_counts.plot(kind='bar', color='skyblue')
    plt.title(f'Top 10 Cities in {state} by Customer Count')
    plt.xlabel('City')
    plt.ylabel('Customer Count')
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```



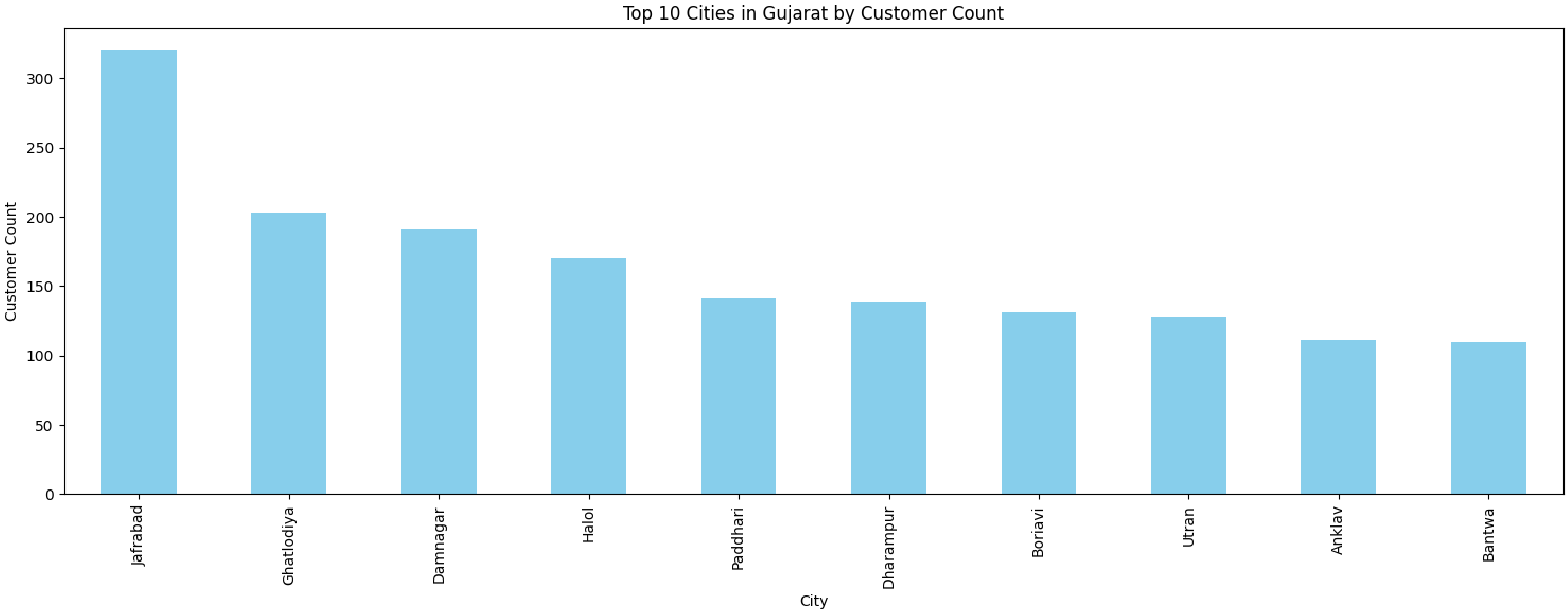
Top 10 Cities in Chhattisgarh by Customer Count



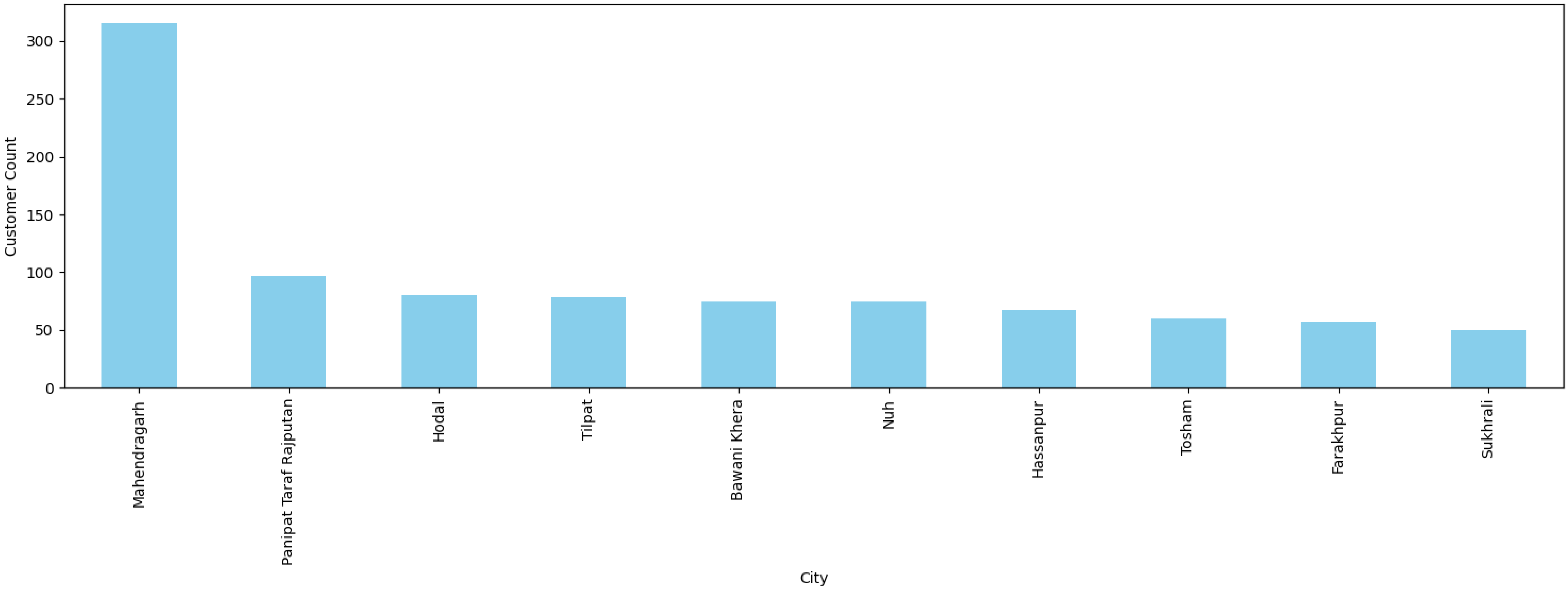
Top 10 Cities in Delhi by Customer Count



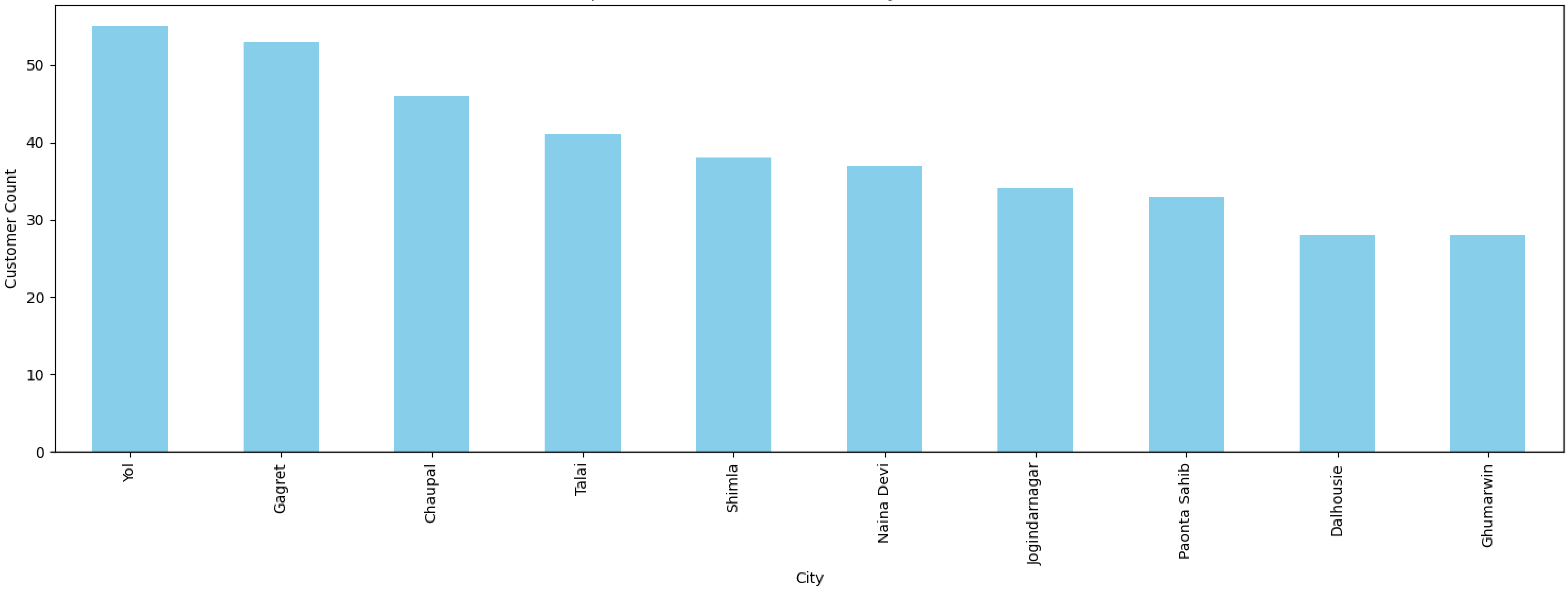
Top 10 Cities in Gujarat by Customer Count



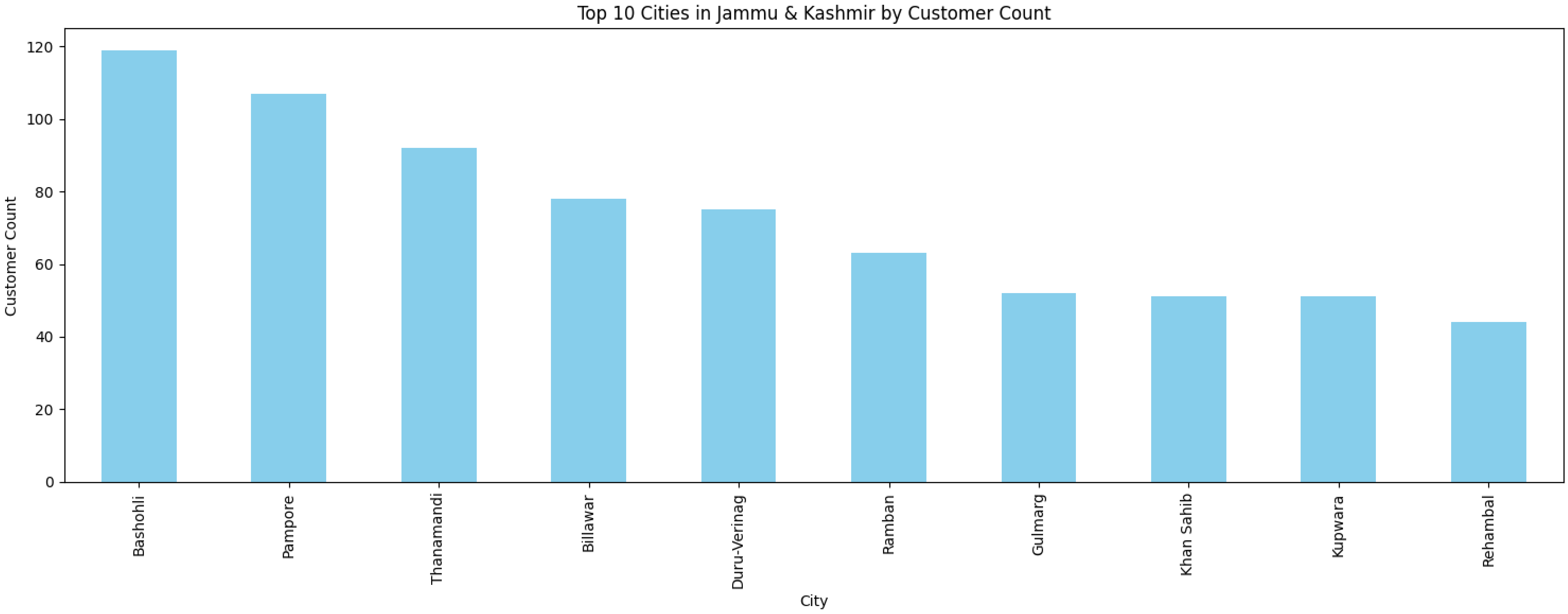
Top 10 Cities in Haryana by Customer Count



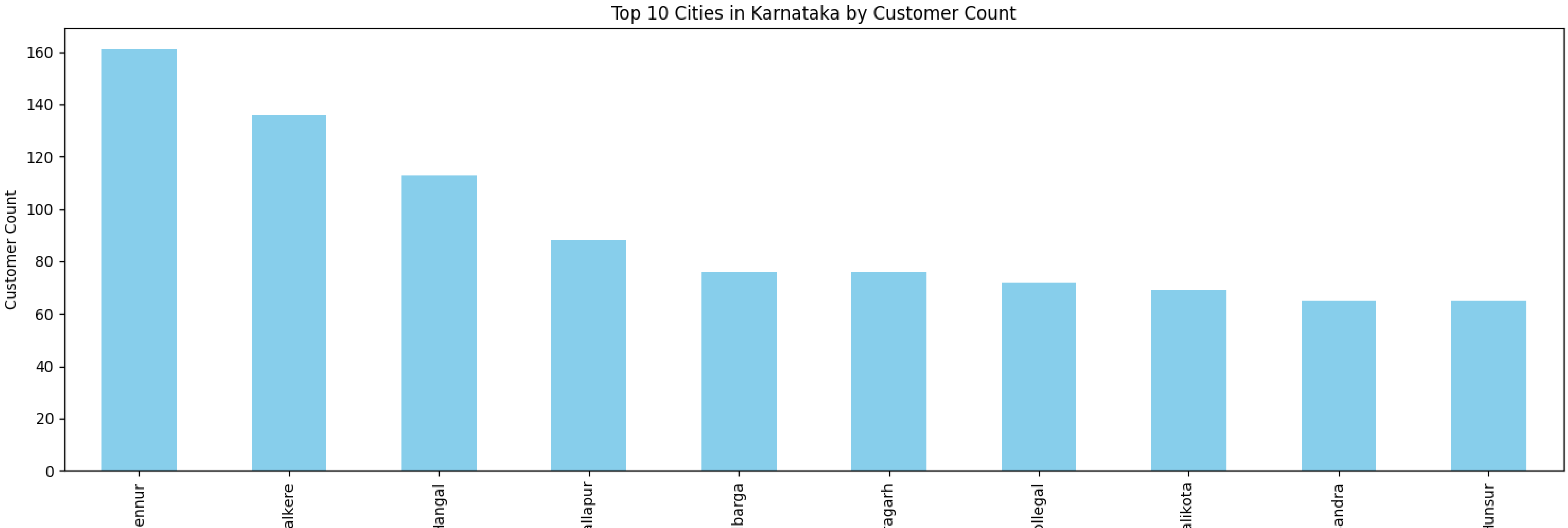
Top 10 Cities in Himachal Pradesh by Customer Count



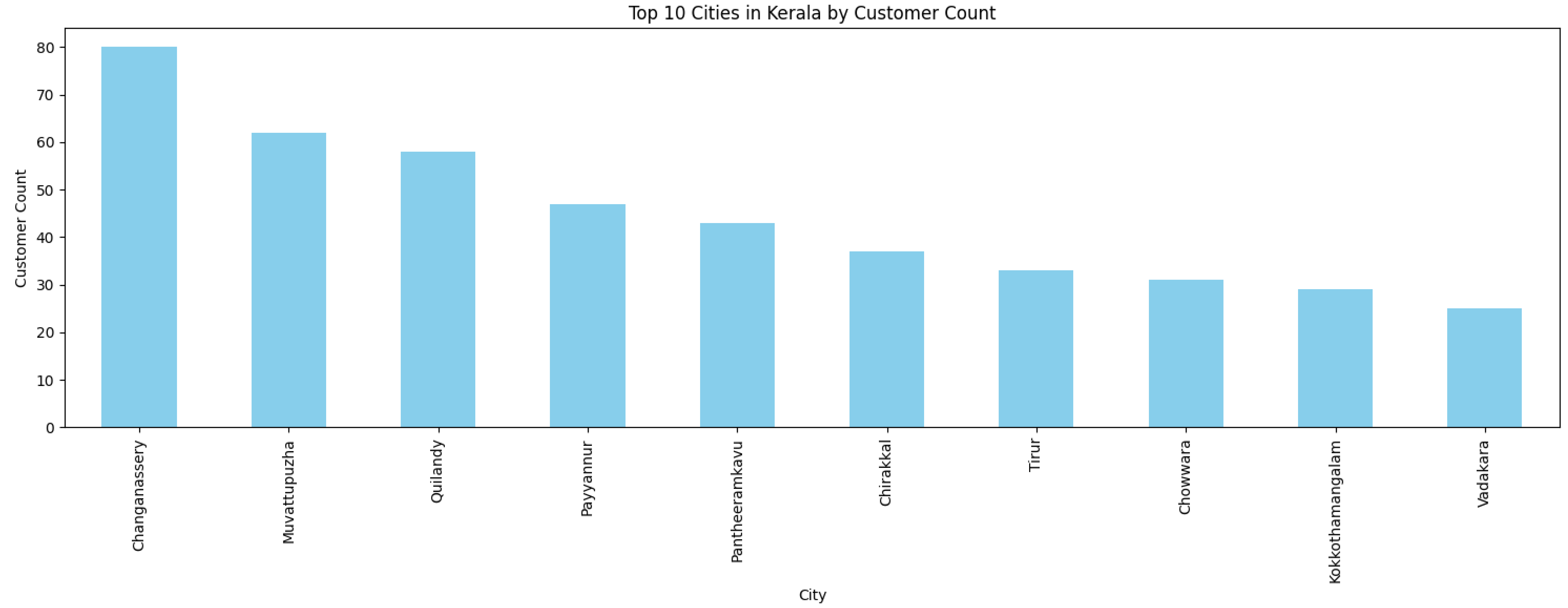
Top 10 Cities in Jammu & Kashmir by Customer Count

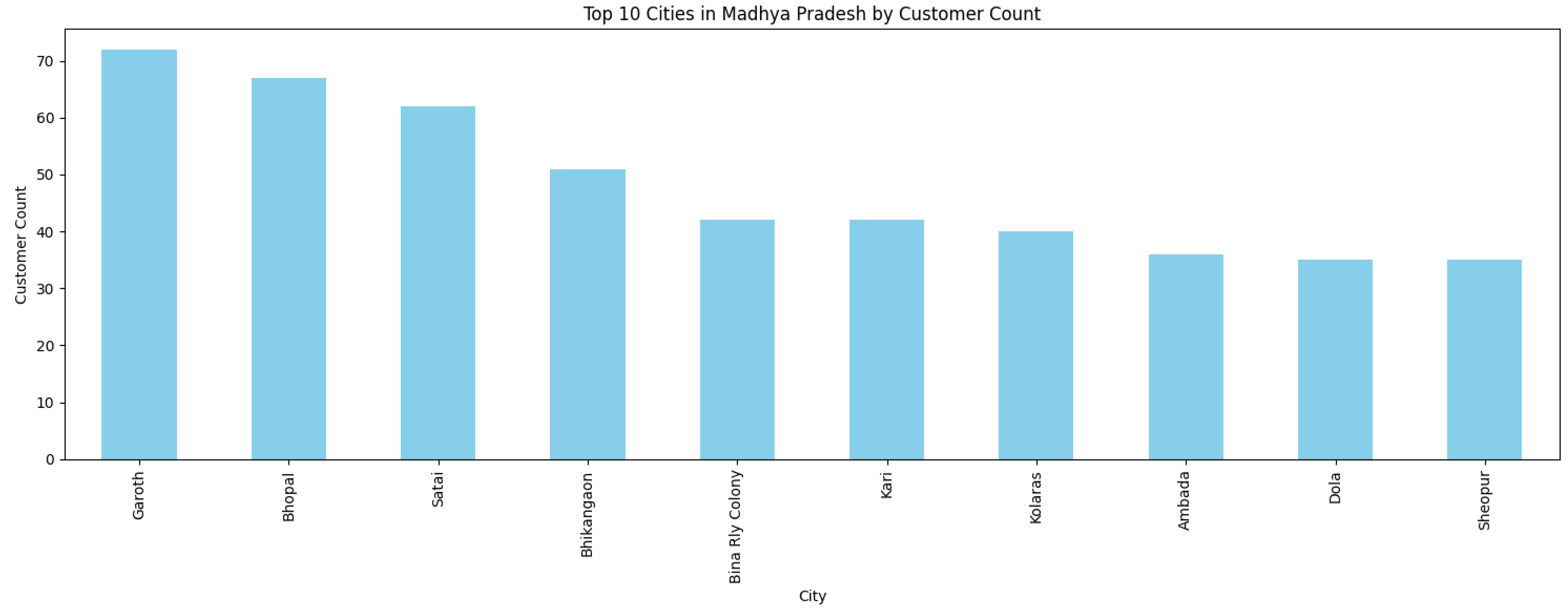


Top 10 Cities in Karnataka by Customer Count

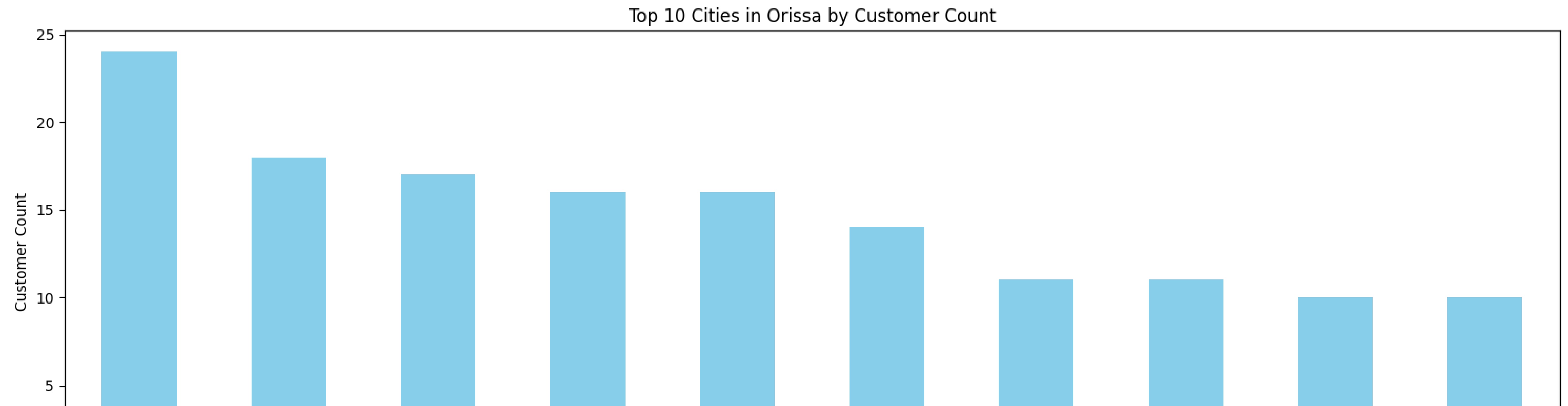


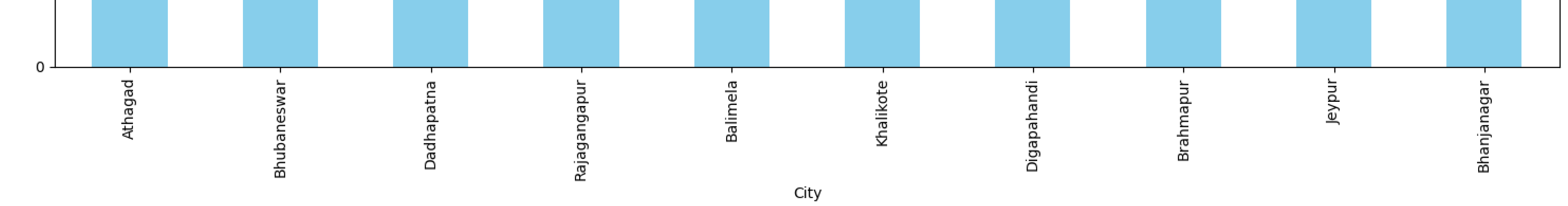

City

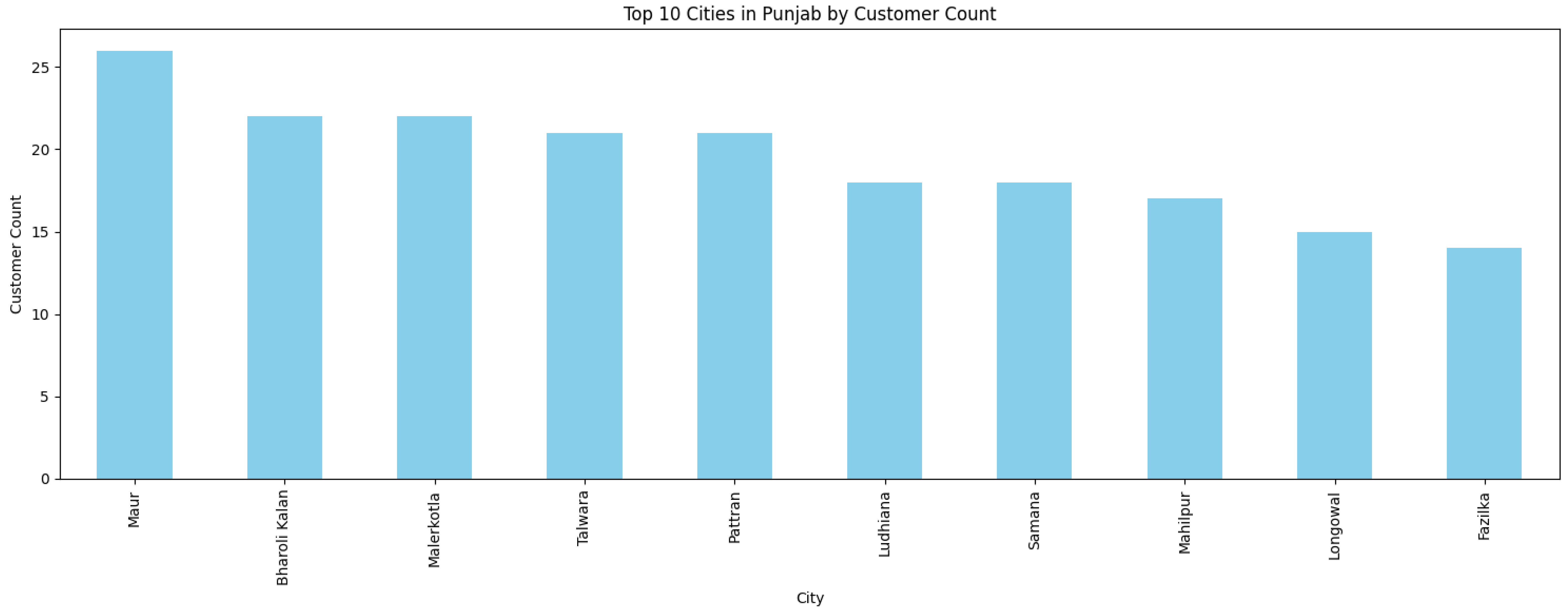

Top 10 Cities in Kerala by Customer Count
Customer Count
0
10
20
30
40
50
60
70
80
Changanassery
Muvattupuzha
Quilandy
Payyannur
Pantheeramkavu
Chirakkal
Tirur
Chowwara
Kokkothamangalam
Vadakara
City

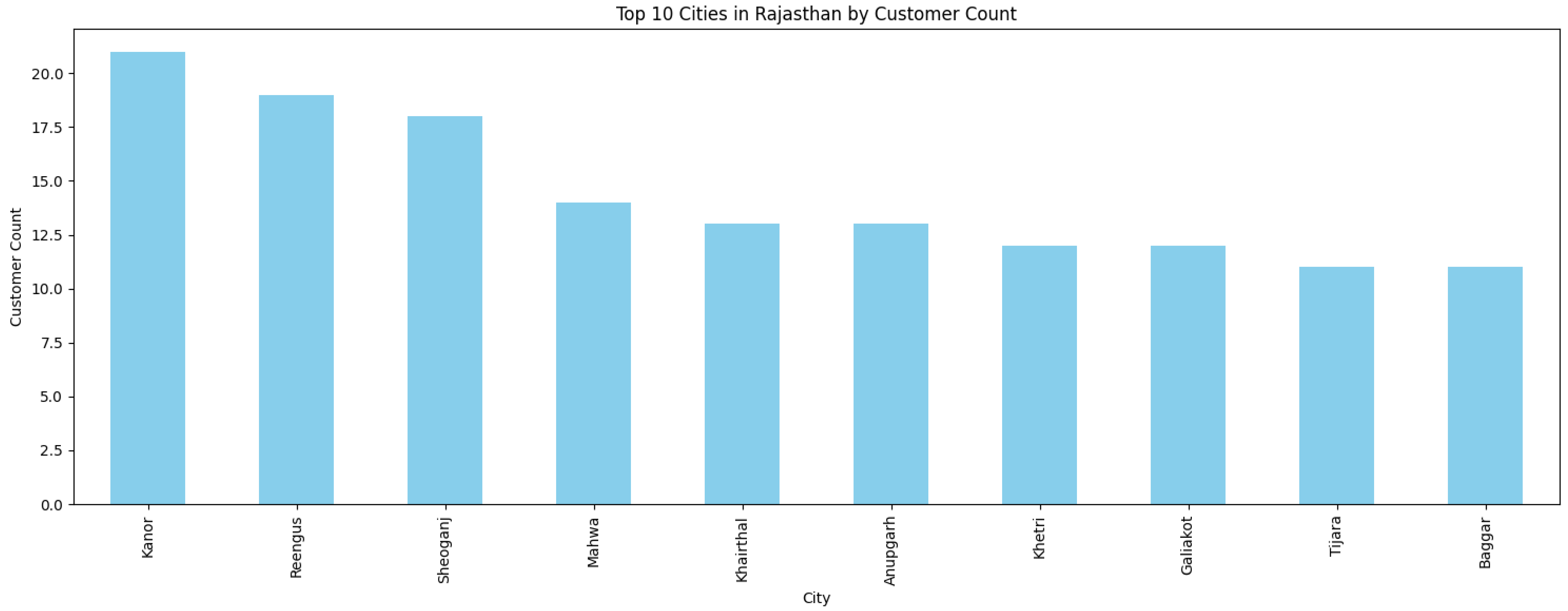

Top 10 Cities in Madhya Pradesh by Customer Count
Customer Count
0
10
20
30
40
50
60
70
Garoth
Bhopal
Satai
Bhikangaon
Bina Rly Colony
Kari
Kolaras
Ambada
Dola
Sheopur
City

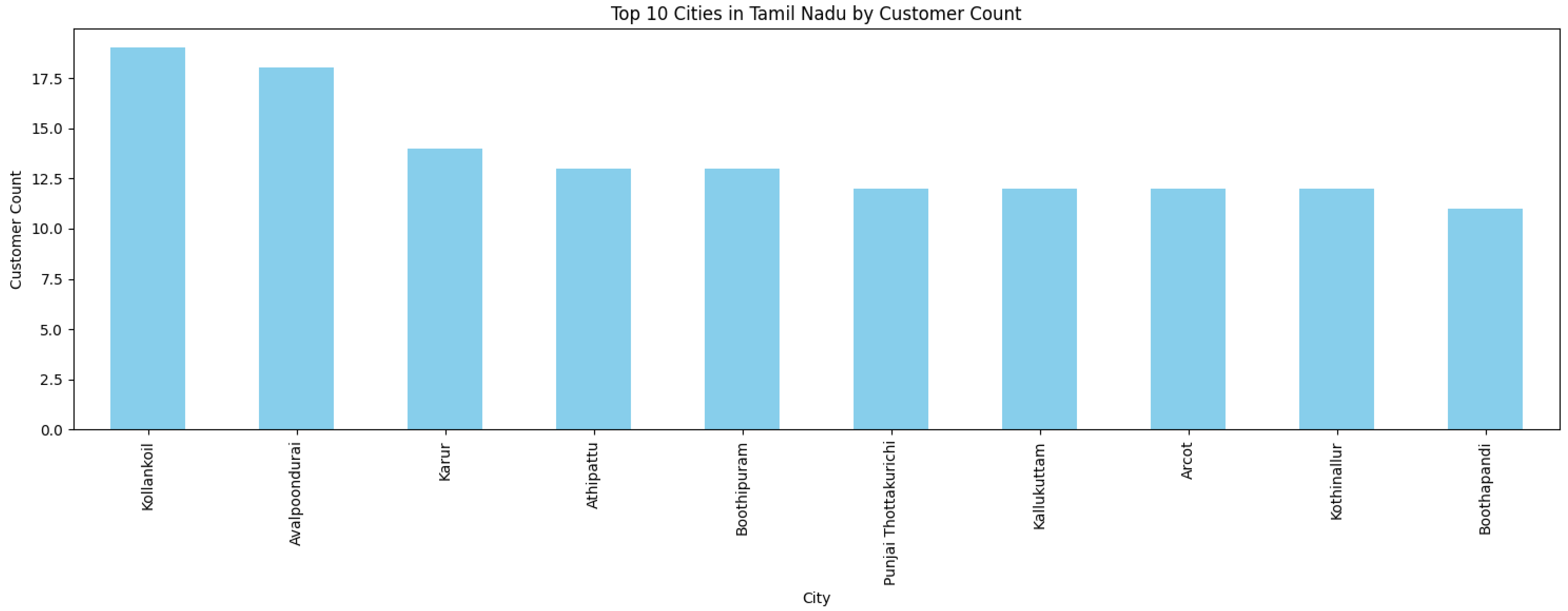

Top 10 Cities in Maharashtra by Customer Count
Customer Count
0
5
10
15
20
25
30
Manjlegaon
Bhiwandi
Nala Sopara
Hadgaon
Kandhar
Dyane
Jintur
Shrirampur
Karanje Turf Satara
Dharmabad
City

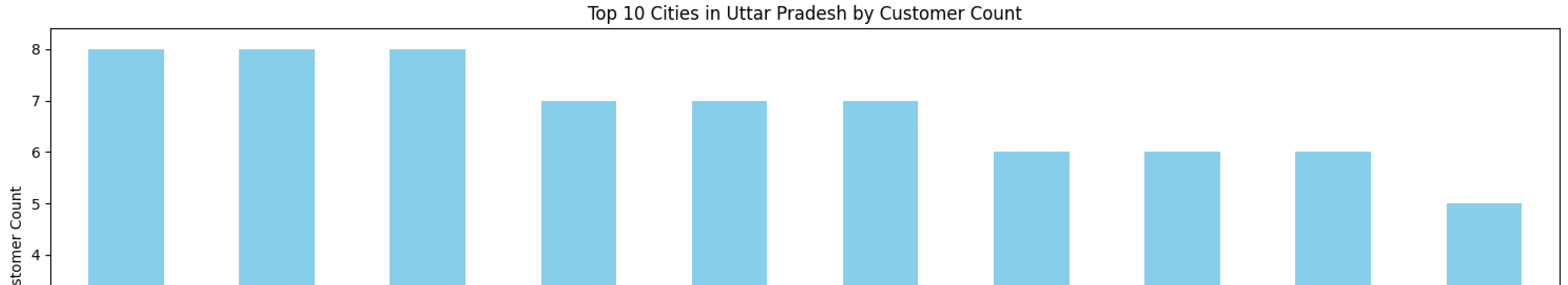

Top 10 Cities in Orissa by Customer Count
Customer Count
5
10
15
20
25

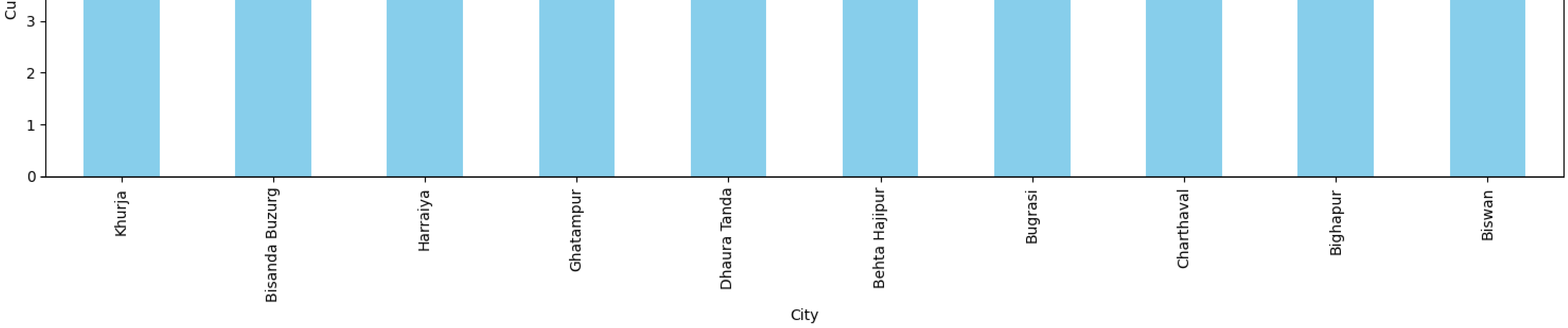

0
Athagad
Bhubaneswar
Dadhapatna
Rajagangapur
Balimela
Khalikote
Digapahandi
Brahmapur
Jeypur
Bhanjanagar
City


Top 10 Cities in Punjab by Customer Count
Customer Count
0
5
10
15
20
25
Maur
Bharoli Kalan
Malerkotla
Talwara
Pattran
Ludhiana
Samana
Mahilpur
Longowal
Fazilka
City

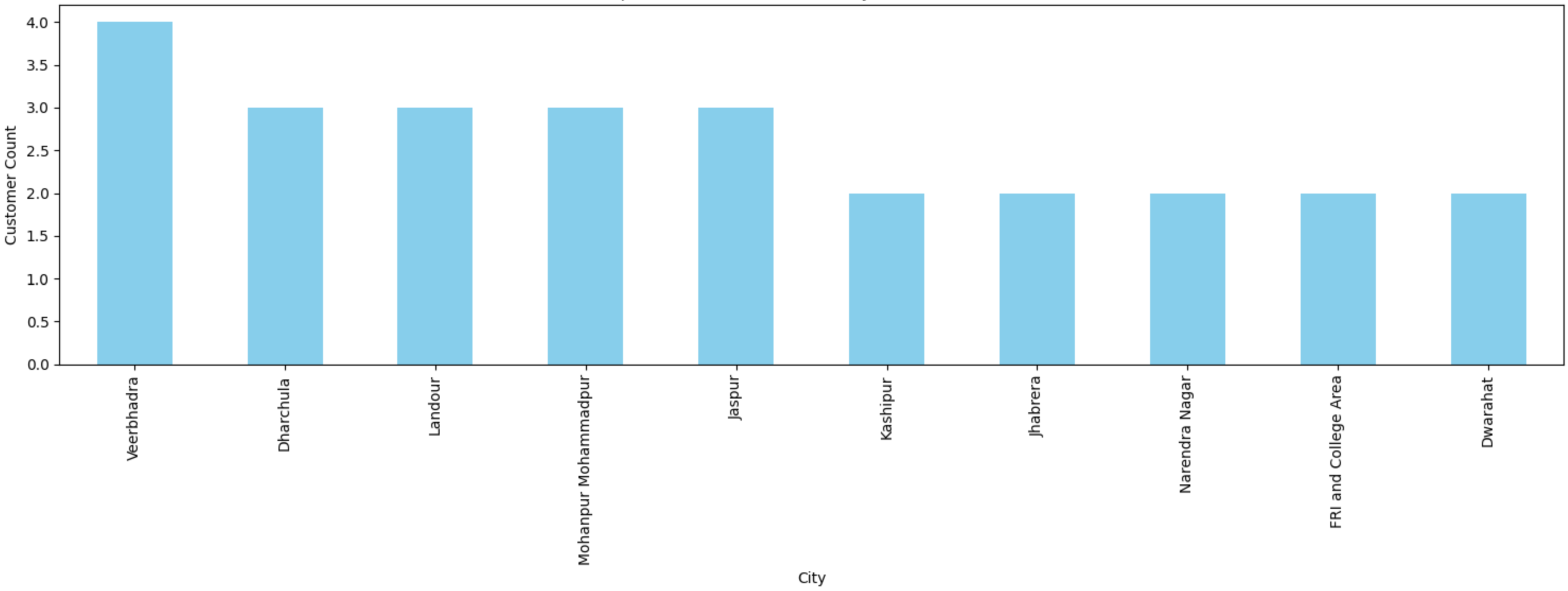

Top 10 Cities in Rajasthan by Customer Count
Customer Count
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Kanor
Reengus
Sheoganj
Mahwa
Khairthal
Anupgarh
Khetri
Galiakot
Tijara
Baggar
City


Top 10 Cities in Tamil Nadu by Customer Count
Customer Count
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Kollankoil
Avalpoondurai
Karur
Athipattu
Boothipuram
Punjai Thottakurichi
Kallukuttam
Arcot
Kothinallur
Boothapandi
City

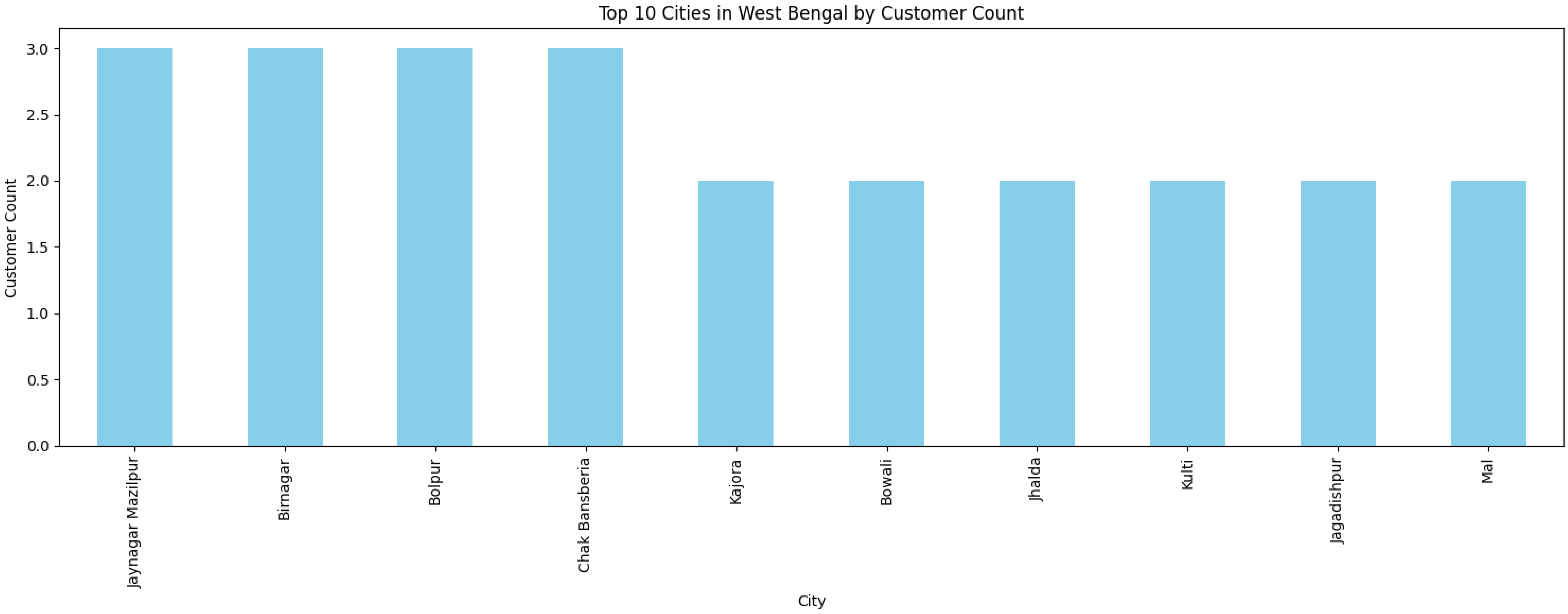

Top 10 Cities in Uttar Pradesh by Customer Count
stomer Count
4
5
6
7
8



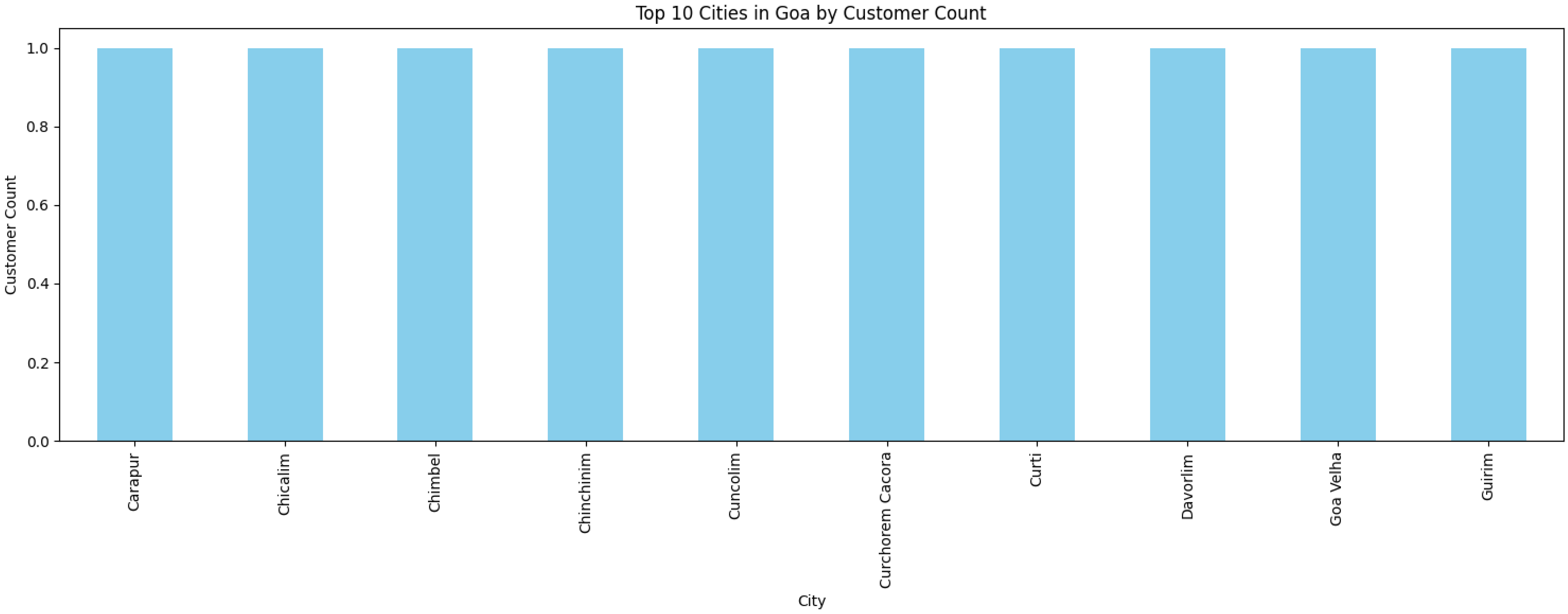
Top 10 Cities in Uttaranchal by Customer Count



Top 10 Cities in West Bengal by Customer Count



Top 10 Cities in Goa by Customer Count



Insights from above chart:

- By visualizing the top cities in each state, can observe the distribution of customers across different regions.This provides insights into the geographic spread of the customer base.
- The cities with the highest customer counts within each state likely represent major population centers or economic hubs. Cities with higher customer counts may indicate stronger competition from local or national competitors in the online marketplace sector.

```
geo_location_data.isnull().sum()
```

```
geolocation_zip_code_prefix    0
geolocation_lat                0
geolocation_lng                0
geolocation_city               0
geolocation_state              0
dtype: int64
```

We have no any null and duplicate value for the geo_location_data dataset, and in above charts we have analyzed for states and cities.

```
orders_data.head()
```

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_es
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	10/2/2017 10:56	10/2/2017 11:07	10/4/2017 19:55	10/10/2017 21:25	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	7/24/2018 20:41	7/26/2018 3:24	7/26/2018 14:31	8/7/2018 15:27	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	8/8/2018 8:38	8/8/2018 8:55	8/8/2018 13:50	8/17/2018 18:06	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	11/18/2017 19:28	11/18/2017 19:45	11/22/2017 13:39	12/2/2017 0:28	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2/13/2018 21:18	2/13/2018 22:20	2/14/2018 19:46	2/16/2018 18:17	

Next steps: Generate code with `orders_data` View recommended plots

```
orders_data.isnull().sum()
```

```
order_id                         0
customer_id                      0
order_status                     0
order_purchase_timestamp         0
order_approved_at                0
order_delivered_carrier_date     0
order_delivered_customer_date    0
order_estimated_delivery_date    0
dtype: int64
```

```
orders_data.duplicated().sum()
```

```
0
```

We have no any null and duplicate value for the orders_data dataset.

```
orders_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96455 entries, 0 to 96454
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       96455 non-null  object
 1   customer_id                    96455 non-null  object
 2   order_status                   96455 non-null  object
 3   order_purchase_timestamp       96455 non-null  object
 4   order_approved_at              96455 non-null  object
 5   order_delivered_carrier_date   96455 non-null  object
 6   order_delivered_customer_date  96455 non-null  object
 7   order_estimated_delivery_date  96455 non-null  object
dtypes: object(8)
memory usage: 5.9+ MB
```

```
orders_data['order_purchase_timestamp'] = pd.to_datetime(orders_data['order_purchase_timestamp'])
```

```
orders_data['order_approved_at'] = pd.to_datetime(orders_data['order_approved_at'])
```

```
orders_data['order_delivered_carrier_date'] = pd.to_datetime(orders_data['order_delivered_carrier_date'])
```

```
orders_data['order_delivered_customer_date'] = pd.to_datetime(orders_data['order_delivered_customer_date'])
```

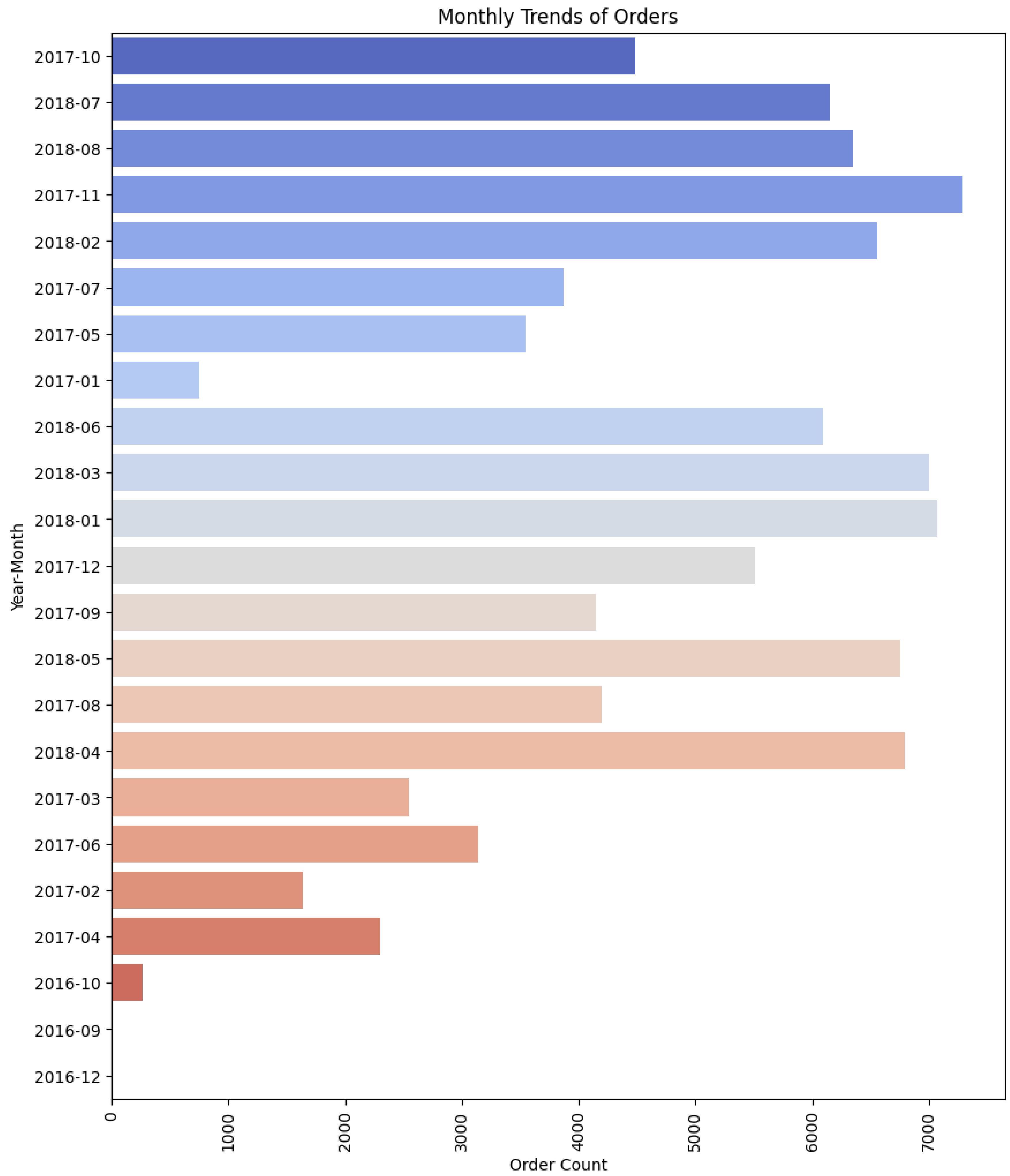
```
orders_data['order_estimated_delivery_date'] = pd.to_datetime(orders_data['order_estimated_delivery_date'])
```

Monthly Treds of order

```
orders_data['Order_Month_year'] = orders_data['order_purchase_timestamp'].dt.strftime('%Y-%m')
```

```
plt.figure(figsize=(10,12))
palette = "coolwarm"
sns.countplot(data=orders_data,y='Order_Month_year',palette=palette)
plt.xlabel('Order Count')
plt.ylabel('Year-Month')
plt.title('Monthly Trends of Orders')
plt.xticks(rotation=90)

plt.show()
```

```
<ipython-input-211-8bf545538f12>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=orders_data,y='Order_Month_year',palette=palette)
```



Insights from above chart:

- The number of orders shows a fluctuating trend over time, with some months having higher order counts compared to others
- There might be seasonal variations in order counts, with certain months experiencing higher order volumes, possibly due to factors like holidays, promotions, or seasonal trends in consumer behavior.**There appears to be a peak in order counts around November 2017, followed by a slight decrease in December and January before another peak in March and April.**

Order approved purchse Difference

```
orders_data['purchase_approved_difference'] = (orders_data['order_approved_at'] - orders_data['order_purchase_timestamp'])

orders_data.head()
```

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_es
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:00	2017-10-02 11:07:00	2017-10-04 19:55:00	2017-10-10 21:25:00	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:00	2018-07-26 03:24:00	2018-07-26 14:31:00	2018-08-07 15:27:00	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:00	2018-08-08 08:55:00	2018-08-08 13:50:00	2018-08-17 18:06:00	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:00	2017-11-18 19:45:00	2017-11-22 13:39:00	2017-12-02 00:28:00	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:00	2018-02-13 22:20:00	2018-02-14 19:46:00	2018-02-16 18:17:00	

Next steps: Generate code with `orders_data` View recommended plots

```
orders_data['purchase_approved_difference'].mean()

    Timedelta('0 days 10:16:42.672748950')
```

0 days indicates that, on average, orders are approved on the same day they are placed.**10:16:42.672748950** indicates that, on average, it takes approximately 10 hours, 16 minutes, and 42 seconds for an order to be approved after it's been placed.

- The fact that orders are typically approved on the same day they are placed suggests that the approval process is relatively efficient. Customers do not have to wait for extended periods for their orders to be approved, which can positively impact their overall satisfaction with the service.
- While the average approval time is relatively short, there may still be room for improvement.

Top 20 Most delayed orders

```
orders_data['purchase_approved_difference'] = (orders_data['order_approved_at'] - orders_data['order_purchase_timestamp'])

orders_data.head()

Purchase_approved_df = orders_data[['order_id','order_purchase_timestamp','order_delivered_carrier_date','purchase_approved_difference']]

max_purchase_approved = Purchase_approved_df.groupby('order_id')['purchase_approved_difference'].mean()

max_purchase_approved_sorted = max_purchase_approved.sort_values(ascending=False).head(20)

max_purchase_approved_sorted
```

```
order_id
0a93b40850d3f4becf2f276666e01340   30 days 21:27:00
f7923db0430587601c2aef15ec4b8af4   30 days 18:27:00
de0076b42a023f53b398ce9ab0d9009c   23 days 02:48:00
daed0f3aefd193de33c31e21b16a3b3a   16 days 20:14:00
9c038e10f14d12a96939a0176c4ecc99   13 days 07:32:00
14ef2221cc3570aa6ce512fc353529b3   13 days 01:48:00
0c1426109d8295a688ee4182016bba59   12 days 12:26:00
483b53ea654d3566427a092cdef047fd   12 days 11:31:00
f5194ba2a4560ffa0e87746852c61fc1   12 days 10:57:00
70f357cca87c1162357bf3c0a993cbe5   12 days 03:24:00
cf72398d0690f841271b695bbfda82d2   12 days 03:19:00
8554cb37f7158cb0b082a841d24a4589   12 days 03:18:00
06eb87385425e5797a1a5c2cdb1b6559   12 days 03:17:00
77ca435b03fbf991e5027e3776e37885   12 days 03:07:00
1fab4ac9d85079b3da72a11475ae1685   12 days 03:02:00
0184d4ddb259e1a4cfc2871888cf97b8   12 days 02:13:00
c3b8c17ee15e0e798c2e178b7d4c7f42   12 days 02:04:00
bc4854efd86d9f42140c951c595d20c1   12 days 01:55:00
40de47dfa620d667117e4a6067b6e1ec   12 days 01:53:00
580f3268bc2075c5961b2d2929c7a35b   12 days 01:46:00
Name: purchase_approved_difference, dtype: timedelta64[ns]
```

Top 20 Most Earliest processed orders

```
max_purchase_approved_sorted = max_purchase_approved.sort_values(ascending=True).head(20)

max_purchase_approved_sorted
```

```
order_id
c82c811a142105c1a90a09d4fb30da62   0 days
0307d42d9c781d32f86cdf1fa5c8670b   0 days
ef69d6f02be07ca6809eb8262e293157   0 days
988a21449c85e5aab93c446596246149   0 days
575b1b9f37a11b59ac98672fa8c487b2   0 days
98c012340047030b894d118bb56be94f   0 days
12cd98b885d11fa8260badce51a54ba0   0 days
5718fc130d653929872f0481438d3f66   0 days
9903b060ffcaca9244088e1f834fde2e   0 days
d8d8abd9a4aa1b36575678bdf5e9d23e   0 days
56c26c7e1d2d600fc5e6a4a6b0be3aa8   0 days
1f308a202c015c950d2a8562bcad7903   0 days
9934e7ff8eaacba82131dd85b7a0e156   0 days
efb6d19e3c56fa371d59421e45e53a49   0 days
994d36ec7dcfa4437a4560fd76447263   0 days
12b9b63b911cf0a24b137a184c48e4ee   0 days
99beb0ca1e257ea579db2f8e68d0165c   0 days
99e34b586d1fa26ea713021ff647d18d   0 days
1299a512fc464f808332d44a0e73f367   0 days
129538e4bc7308e9ab041540283813f2   0 days
Name: purchase_approved_difference, dtype: timedelta64[ns]
```

Order approved and order deliverd to customer Difference

```
orders_data['approved_deliverd_difference'] = orders_data['order_delivered_customer_date']-orders_data['order_approved_at']

orders_data['approved_deliverd_difference'].mean()
```

```
Timedelta('12 days 03:06:18.230470167')
```

- **On average, there is a time gap of approximately 12 days, 3 hours, and 6 minutes between when an order is approved and when it is delivered to the customer.**
- A longer average time indicates that inefficiencies or bottlenecks in the order fulfillment process, such as delays in order processing, packaging, shipping, or delivery
- Meeting or exceeding customer expectations regarding order delivery times is crucial for maintaining customer satisfaction and loyalty.

Ensuring orders are completed within Service Level Agreements (SLAs) is essential for maintaining customer satisfaction and meeting business objectives. Here are the necessary steps to achieve this:

1. Streamline order management processes to minimize delays and errors. Implement efficient order processing systems, automate routine tasks, and prioritize orders based on urgency and SLAs.
2. Maintain adequate inventory levels to fulfill orders promptly. Implement inventory tracking systems to monitor stock levels in real-time and prevent stockouts or overstock situations.
3. Partner with reliable shipping carriers or logistics providers to ensure timely and cost-effective order delivery. Negotiate service level agreements with shipping partners to meet SLAs for transit times and delivery reliability.
4. Optimize warehouse and fulfillment operations to expedite order processing and shipping. Implement efficient picking, packing, and shipping processes to reduce turnaround times.
5. Provide customers with real-time order tracking capabilities to monitor the status and location of their orders. Utilize tracking technologies such as GPS and order tracking software to enhance transparency and communication.

Order approved and order deliverd to customer Difference

Top 20 Most delayed deliverd orders

```
approved_deliverd_df = orders_data[['order_id','order_approved_at','order_delivered_customer_date','approved_deliverd_difference']]

approved_deliverd = approved_deliverd_df.groupby('order_id')['approved_deliverd_difference'].mean()

approved_deliverd_sorted =approved_deliverd.sort_values(ascending=False).head(20)

approved_deliverd_sorted
```

```
order_id
ca07593549f1816d26a572e06dc1eab6   208 days 12:01:00
1b3190b2dfa9d789e1f14c05b647a14a   208 days 08:08:00
2fb597c2f772eca01b1f5c561bf6cc7b   194 days 20:24:00
```

```
285ab9426d6982034523a855f55a885e   194 days 15:13:00
440d0d17af552815d15a9e41abe49359   194 days 14:01:00
0f4519c5f1c541ddec9f21b3bddd533a   194 days 01:12:00
47b40429ed8cce3aee9199792275433f   191 days 10:20:00
2fe324febf907e3ea3f2aa9650869fa5   189 days 20:43:00
2d7561026d542c8dbd8f0daeadf67a43   188 days 03:14:00
c27815f7e3dd0b926b58552628481575   187 days 17:51:00
437222e3fd1b07396f1d9ba8c15fba59   187 days 04:52:00
dfe5f68118c2576143240b8d78e5940a   186 days 05:41:00
6e82dcfb5eada6283dba34f164e636f5   181 days 15:34:00
2ba1366baecad3c3536f27546d129017   179 days 12:28:00
d24e8541128cea179a11a65176e0a96f   175 days 05:11:00
3566eabb132f8d64741ae7b921bbd10e   173 days 12:57:00
2fa29503f2ebd9f53deba187160f3202   172 days 03:09:00
ed8e9faf1b75f43ee027103957135663   171 days 06:51:00
df6d8b7768a047c2981bae0a24afbb01   168 days 04:13:00
525e11b26fdb7f41471d289897d0f6da   167 days 22:28:00
Name: approved_deliverd_difference, dtype: timedelta64[ns]
```

Above are the top 20 most delayed orders for delivery after order approval

```
approved_deliverd_sorted =approved_deliverd.sort_values(ascending=True).head(20)
```

```
approved_deliverd_sorted
```

```
order_id
bc4854efd86d9f42140c951c595d20c1   -7 days +00:15:00
1fab4ac9d85079b3da72a11475ae1685   -6 days +22:07:00
8554cb37f7158cb0b082a841d24a4589   -6 days +22:09:00
40de47dfa620d667117e4a6067b6e1ec   -6 days +22:43:00
4387477eec4b3c89b39f3f454940d059   -6 days +23:00:00
e73fe43cdcd166f7f0c6e3c2bf11a917   -6 days +23:49:00
6dcf0aeb8b1eb4021c26e1d0e9394979   -5 days +03:07:00
6b80bb20190715d71c43efff617bd659   -5 days +03:36:00
0184d4ddb259e1a4cfc2871888cf97b8   -5 days +16:55:00
bc1b85147b5edbb7cbefcf5c1bd5ded9   -5 days +21:26:00
a727355acb88d9b3e6e41fb2e3888a0e   -4 days +18:04:00
f222c56f035b47dfa1e069a88235d730   -4 days +20:39:00
6e57e23ecac1ae881286657694444267   -3 days +00:50:00
6fa0c125ee7d870f6602c97e33d87bc5   -3 days +04:52:00
66e1b657a71397245290f39ffe24031e   -3 days +15:27:00
cf72398d0690f841271b695bbfda82d2   -3 days +16:11:00
6df6c9c9af6ef75b4f06f8a7b9f47e9c   -3 days +17:32:00
c3b8c17ee15e0e798c2e178b7d4c7f42   -3 days +18:35:00
d836abb4444d8594455e9766104e958c   -3 days +18:38:00
fa962e76e50f3469ae2abfa54e6d5be0   -3 days +18:51:00
Name: approved_deliverd_difference, dtype: timedelta64[ns]
```

The negative average differences suggest potential data quality issues or errors in the timestamps recorded for order approval and delivery. It's essential to review and verify the accuracy of the timestamps to ensure data integrity and reliability.

Order delivered vs Estimated Delivery

```
orders_data['estimated_vs_actual_delivery'] = orders_data['order_estimated_delivery_date']-orders_data['order_delivered_customer_date']
```

```
orders_data['estimated_vs_actual_delivery'].mean()
```

```
Timedelta('11 days 04:15:08.395210201')
```

- **On average, there is a delay of approximately 11 days from the estimated delivery date to the actual delivery date. This suggests that, on average, orders are delivered later than initially estimated.**
- Understanding the average delay between estimated and actual delivery dates can help manage customer expectations. Businesses can use this information to provide more accurate estimated delivery dates or to communicate potential delays to customers proactively.
- Timely delivery is often crucial for customer satisfaction. By reducing the average delivery delay, businesses can enhance customer experience and loyalty.

order_items_data

```
order_items_data.isnull().sum()
```

```
order_id               0
order_item_id          0
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64
```

```
order_items_data.duplicated().sum()
```

```
0
```

No any Null and duplicated value present in order_items_data dataset

```
order_payment_data.head()
```

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45

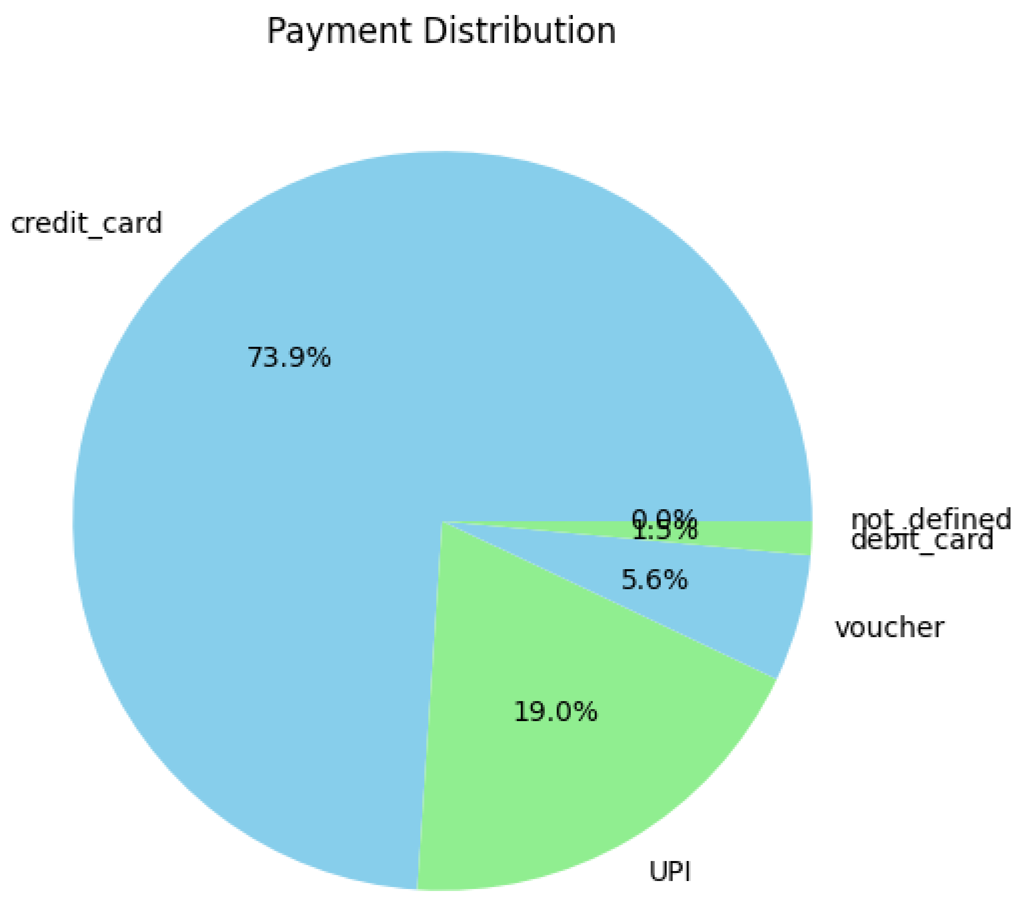
```
order_payment_data.isnull().sum()
```

```
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64
```

Payment type Distribution

```
payment_distribution = order_payment_data['payment_type'].value_counts()

print(payment_distribution)

plt.figure(figsize=(15,6))
plt.pie(payment_distribution.values,labels=payment_distribution.index,autopct='%1.1f%%', colors=['skyblue', 'lightgreen'])
plt.title('Payment Distribution')
```

```
credit_card    76795
UPI            19784
voucher         5775
debit_card      1529
not_defined        3
Name: payment_type, dtype: int64
Text(0.5, 1.0, 'Payment Distribution')
```



- Payment method distribution reflects customer preferences and behavior, providing insights into how customers prefer to complete transactions.
- The significant proportion of transactions made through digital payment methods like credit cards and UPI reflects the increasing trend towards cashless transactions and the adoption of digital payment technologies.
- UPI (Unified Payments Interface) is the second most popular payment method, indicating the growing adoption of digital payment solutions.Voucher and debit card transactions constitute smaller percentages compared to credit cards and UPI.
- The relatively low percentage of transactions using debit cards suggests that there may be opportunities to promote debit card payments or improve the user experience for customers using this payment method.

```
order_payment_data['payment_value'].max()
```

```
13664.08
```

```
order_payment_data['payment_value'].mean()
```

```
154.10038041699553
```

The maximum payment value observed in the dataset is $13,664.08. This represents the highest amount paid for a single transaction. The average payment value across all transactions is approximately$ 154.10

```
order_review_data.isnull().sum()
```

```
review_id                  0
order_id                   0
review_score               0
review_creation_date       0
review_answer_timestamp    0
dtype: int64
```

```
order_review_data.duplicated().sum()
```

```
0
```

```
order_review_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 5 columns):
 #   Column                   Non-Null Count   Dtype 
---  ------                   --------------   ----- 
 0   review_id                100000 non-null  object
 1   order_id                 100000 non-null  object
 2   review_score             100000 non-null  int64 
 3   review_creation_date     100000 non-null  object
 4   review_answer_timestamp  100000 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
```

No any Null values and duplicate value present in this order_review_data dataset

```
product_data.head()
```

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_
0	1e9e8ef04dbcff4541ed26657ea517e5	Perfumery	40	287	1	225	16	10	
1	3aa071139cb16b67ca9e5dea641aaa2f	Art	44	276	1	1000	30	18	
2	96bd76ec8810374ed1b65e291975717f	Sports_Leisure	46	250	1	154	18	9	
3	cef67bcfe19066a932b7673e239eb23d	Baby	27	261	1	371	26	4	
4	9dc1a7de274444849c219cff195d0b71	Housewares	37	402	4	625	20	17	

Next steps: Generate code with product_data View recommended plots

```
product_data.isnull().sum()
```

```
product_id                    0
product_category_name         0
product_name_lenght           0
product_description_lenght    0
product_photos_qty            0
product_weight_g              0
product_length_cm             0
product_height_cm             0
product_width_cm              0
dtype: int64
```

```
product_data.duplicated().sum()
```

```
0
```

No any Null values and duplicate value present in this product_data dataset

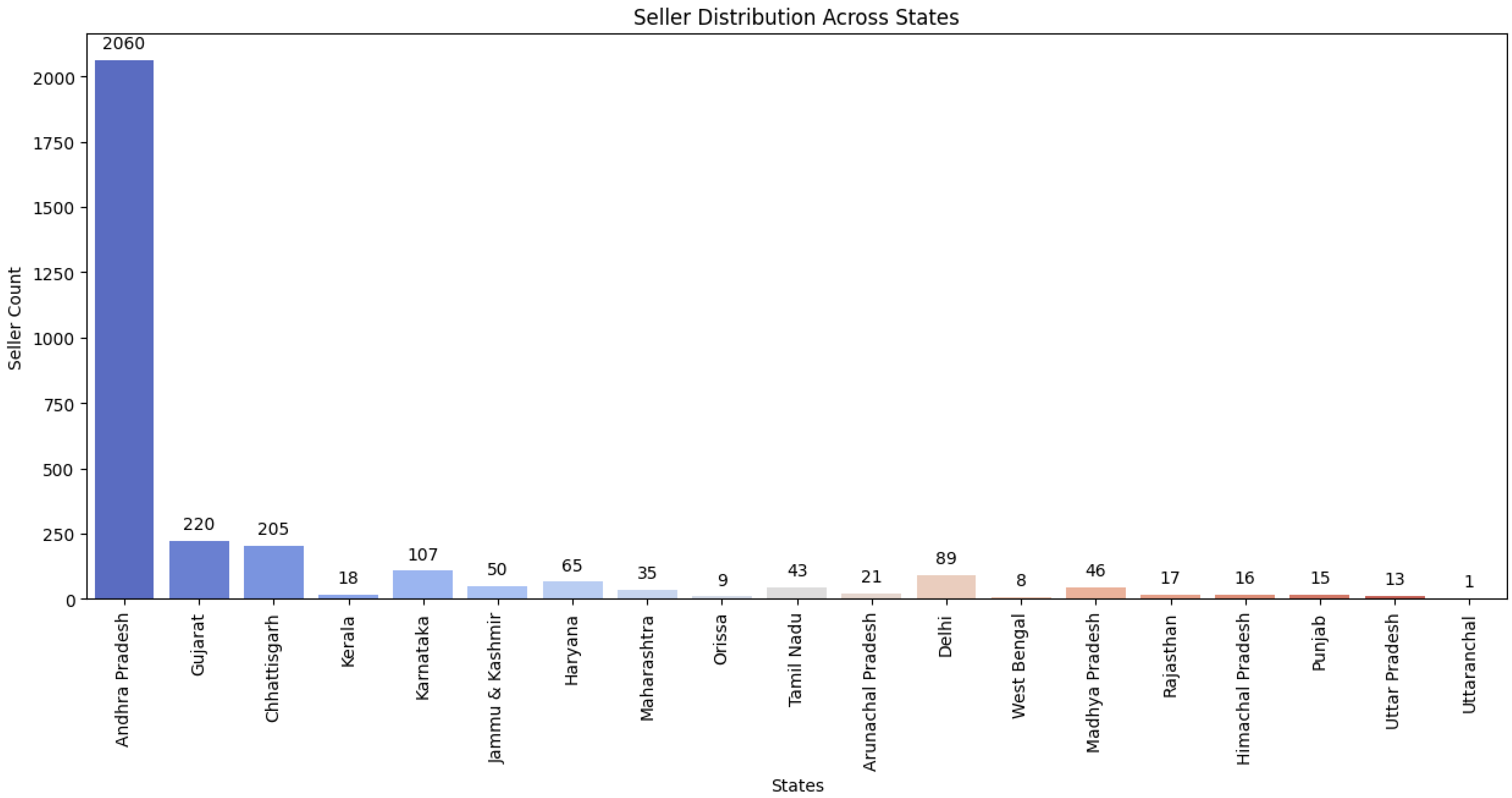
```
sellers_data.head()
```

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	Alwal	Andhra Pradesh
1	d1b65fc7debc3361ea86b5f14c68d2e2	13023	Alwal	Andhra Pradesh
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	Badepalle	Andhra Pradesh
3	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	Akkarampalle	Andhra Pradesh
4	51a04a8a6bdcb23deccc82b0b80742cf	12914	Koratla	Andhra Pradesh

Next steps: Generate code with sellers_data View recommended plots

```
seller_counts = sellers_data['seller_state'].value_counts()

print(seller_counts)

plt.figure(figsize=(15, 6))
palette = "coolwarm"
ax = sns.countplot(x='seller_state', data=sellers_data, palette=palette)

# Adding data labels
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, 10),
                textcoords = 'offset points')


plt.xlabel('States')
plt.ylabel('Seller Count')
plt.title('Seller Distribution Across States')
plt.xticks(rotation=90)
plt.show()
```

```
Andhra Pradesh       2060
Gujarat               220
Chhattisgarh          205
Karnataka             107
Delhi                  89
Haryana                65
Jammu & Kashmir        50
Madhya Pradesh         46
Tamil Nadu             43
Maharashtra            35
Arunachal Pradesh      21
Kerala                 18
Rajasthan              17
Himachal Pradesh       16
Punjab                 15
Uttar Pradesh          13
Orissa                  9
West Bengal             8
Uttaranchal             1
Name: seller_state, dtype: int64
<ipython-input-234-461db3133450>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(x='seller_state', data=sellers_data, palette=palette)
```



- States like Andhra Pradesh, Gujarat, and Chhattisgarh have a relatively higher number of sellers, indicating potential business activity or a concentration of sellers in those regions.
- In contrast, states like Uttaranchal, West Bengal, and Orissa have fewer sellers, suggesting lower participation or presence of sellers in those areas.

Now will move towards Business Objects

Define & calculate high level metrics like (Total Revenue, Total quantity, Total products, Total categories, Total sellers, Total locations, Total channels, Totalpayment methods etc...)

Total Revenue

```
order_payment_data['Revenue'] = order_payment_data['payment_installments'] * order_payment_data['payment_value']

order_payment_data.head()

Total_Revenue = order_payment_data['Revenue'].sum()

print('Total Revenue = ',Total_Revenue)
```

```
Total Revenue =  65763157.62
```

```
Total_Quantity = order_items_data['order_item_id'].sum()

print('Total Quantity = ',Total_Quantity)
```

```
Total Quantity =  134936
```

```
Total_Product = product_data['product_id'].nunique()

print('Total Product = ',Total_Product)
```

```
Total Product =  32327
```

```
Total_Categories = product_data['product_category_name'].nunique()

print('Total Product Categories = ',Total_Categories)
```

```
Total Product Categories =  71
```

```
Total_sellers = sellers_data['seller_id'].nunique()

print('Total Seller =',Total_sellers)
```

```
Total Seller = 3095
```

```
Total_Location = geo_location_data['geolocation_zip_code_prefix'].nunique()

print('Total Location =',Total_Location)

Total_States = geo_location_data['geolocation_state'].nunique()

print('Total States =',Total_States)

Total_cities = geo_location_data['geolocation_city'].nunique()

print('Total Cities =',Total_cities)
```

```
Total Location = 19015
Total States = 20
Total Cities = 3809
```

```
Total_channel = order_payment_data['payment_type'].nunique()

print('Total Payment Channel = ',Total_channel)
```

```
Total Payment Channel =  5
```

- **Total Revenue:** The total revenue generated during the specified period is $65,763,157.62. This indicates the overall financial performance of the business over the given timeframe.
- **Total Quantity:** A total of 134,936 products were sold. This metric provides insights into the volume of goods moved through the marketplace.
- **Total Products:** The marketplace offers a wide range of products, with a total of 32,327 unique products available. This suggests diversity in product offerings, catering to various customer preferences.
- **Total Product Categories:** There are 71 distinct product categories available on the platform. Understanding the distribution of products across these categories can help identify popular and niche segments.
- **Total Sellers:** The platform is supported by 3,095 sellers. A large number of sellers indicate a vibrant ecosystem of suppliers contributing to the marketplace.
- **Total Locations, States, and Cities:** The marketplace operates across 19,015 locations, spanning 20 states and 3,809 cities in India. This extensive geographic coverage suggests a wide reach and potential customer base.
- **Total Payment Channels:** There are 5 payment channels available for customers to complete transactions. Analyzing payment preferences and channel usage can provide insights into customer behavior and preferences.

Understanding how many new customers acquired every month

```
customer_order_data = pd.merge(customers_data,orders_data,how = 'left',on ='customer_id')

customer_order_data.head()
```

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state	order_id	order_status	order_purchase_timesta
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	Adilabad	Andhra Pradesh	00e7ee1b050b8499577073aeb2a297a1	delivered	2017-05-16 15:05
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	Adoni	Andhra Pradesh	29150127e6685892b6eab3eec79f59c7	delivered	2018-01-12 20:48
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	Akkarampalle	Andhra Pradesh	b2059ed67ce144a36e2aa97d2c9e9ad2	delivered	2018-05-19 16:07
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	Akkayapalle	Andhra Pradesh	951670f92359f4fe4a63112aa7306eba	delivered	2018-03-13 16:06
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	Alwal	Andhra Pradesh	6b7d50bd145f6fc7f33cebabd7e49d0f	delivered	2018-07-29 09:51

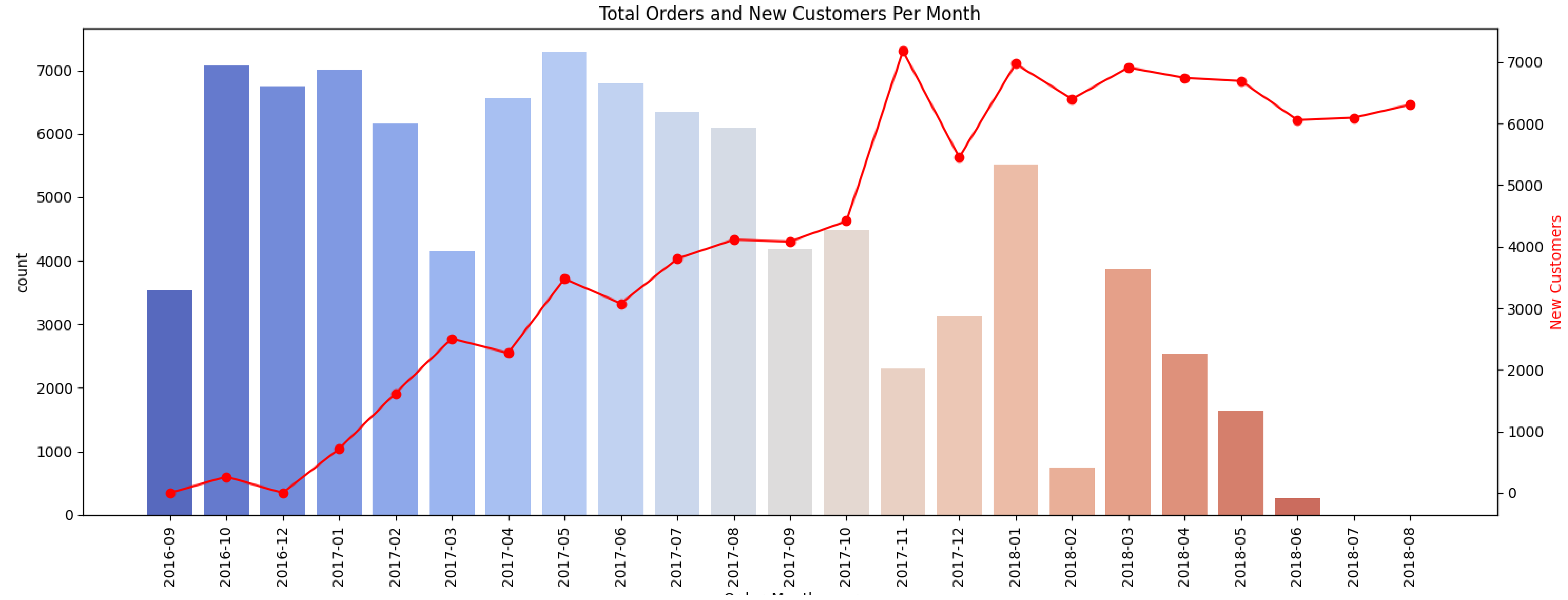
Next steps: Generate code with `customer_order_data` View recommended plots

```
new_customers_per_month = customer_order_data.groupby('Order_Month_year')['customer_unique_id'].nunique()

new_customers_per_month.reset_index()
```

	Order_Month_year	customer_unique_id
0	2016-09	1
1	2016-10	262
2	2016-12	1
3	2017-01	716
4	2017-02	1618
5	2017-03	2508
6	2017-04	2274
7	2017-05	3478
8	2017-06	3076
9	2017-07	3802
10	2017-08	4114
11	2017-09	4082
12	2017-10	4417
13	2017-11	7182
14	2017-12	5450
15	2018-01	6974
16	2018-02	6400
17	2018-03	6914
18	2018-04	6744
19	2018-05	6693
20	2018-06	6058
21	2018-07	6097
22	2018-08	6310

```
plt.figure(figsize=(15, 6))

sns.countplot(data=customer_order_data, x='Order_Month_year', palette='coolwarm')
plt.xticks(rotation=90)

plt.twinx()
plt.plot(new_customers_per_month.index.astype(str), new_customers_per_month.values, color='red', marker='o')
plt.title('Total Orders and New Customers Per Month')
plt.xlabel('Month-Year')
plt.ylabel('Total Orders', color='blue')
plt.ylabel('New Customers', color='red')
plt.tight_layout()
plt.show()
```

```
<ipython-input-244-901288b40309>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=customer_order_data, x='Order_Month_year', palette='coolwarm')
```



Insights from above chart:

- The countplot of total orders per month shows fluctuations in order volume over time. There seems to be variability in the number of orders placed across different months, indicating potential seasonal patterns or changes in customer behavior.
- The line plot of new customers per month overlaid on the countplot highlights the trend in acquiring new customers over the same period.
- Observing the relationship between total orders and new customers per month can provide insights into the effectiveness of marketing campaigns, promotions, or other initiatives aimed at attracting new customers.
- Analyzing the combined plot allows for a holistic evaluation of business performance, considering both customer acquisition and order volume. It helps identify periods of growth, potential challenges, and areas for improvement in customer acquisition and order fulfillment processes.

Understand the retention of customers on month on month basis

```
monthly_customers = customer_order_data.groupby('Order_Month_year')['customer_unique_id'].count().reset_index()
```

```
monthly_customers
```

	Order_Month_year	customer_unique_id
0	2016-09	1
1	2016-10	265
2	2016-12	1
3	2017-01	748
4	2017-02	1641
5	2017-03	2546
6	2017-04	2303
7	2017-05	3545
8	2017-06	3135
9	2017-07	3872
10	2017-08	4193
11	2017-09	4149
12	2017-10	4478
13	2017-11	7288
14	2017-12	5513
15	2018-01	7069
16	2018-02	6555
17	2018-03	7003
18	2018-04	6798
19	2018-05	6749
20	2018-06	6096
21	2018-07	6156
22	2018-08	6351

Next steps: Generate code with monthly_customers | View recommended plots

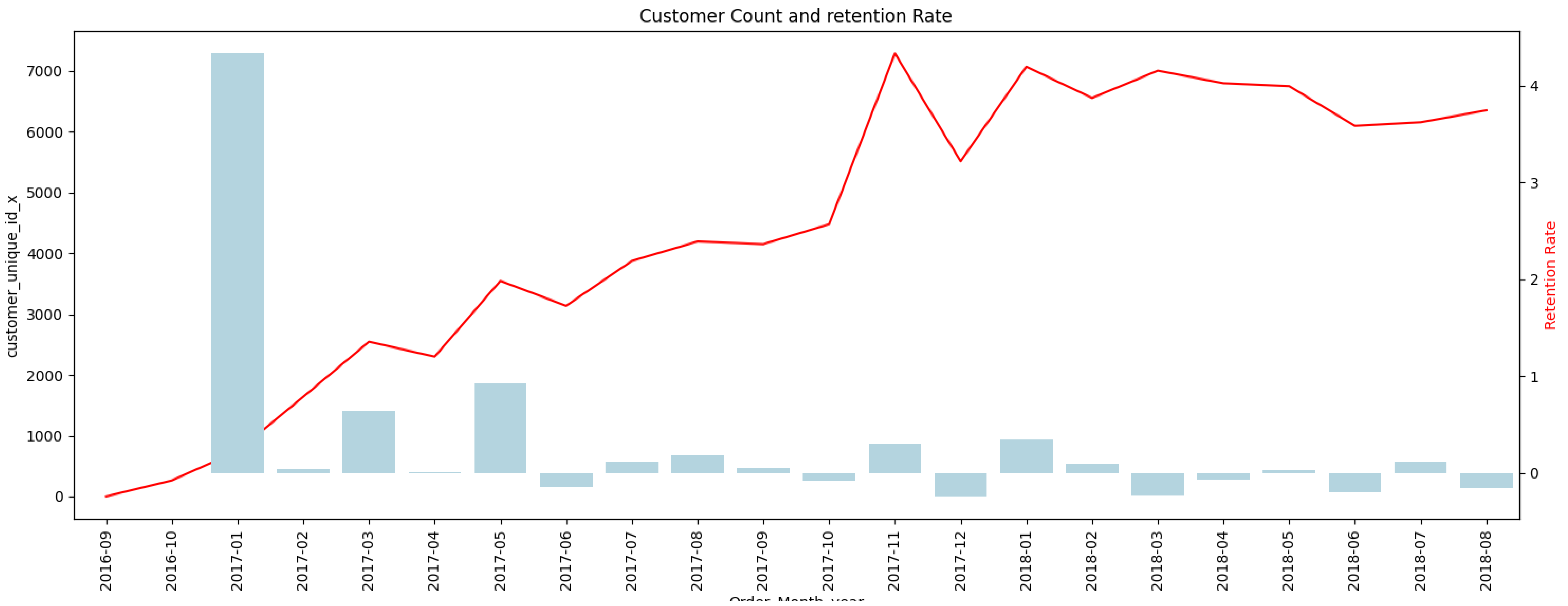
```
Cust_YrMth=customer_order_data.loc[:,['customer_unique_id','Order_Month_year']]

Cust_YrMth.drop_duplicates(subset=['customer_unique_id'],keep='first',inplace=True)

CustAcquired=Cust_YrMth.groupby('Order_Month_year')['customer_unique_id'].count()

CustAcquired.reset_index()
```

	Order_Month_year	customer_unique_id
0	2016-09	1
1	2016-10	256
2	2017-01	700
3	2017-02	1591
4	2017-03	2464
5	2017-04	2220
6	2017-05	3385
7	2017-06	2998
8	2017-07	3718
9	2017-08	4011
10	2017-09	3958
11	2017-10	4302
12	2017-11	7059
13	2017-12	5339
14	2018-01	6834
15	2018-02	6296
16	2018-03	6805
17	2018-04	6613
18	2018-05	6558
19	2018-06	5942
20	2018-07	5984
21	2018-08	6206

```
cust_ret = pd.merge(left=monthly_customers,right=CustAcquired,on='Order_Month_year')

cust_ret

cust_ret['Customer_retention_count'] = cust_ret.customer_unique_id_x - cust_ret.customer_unique_id_y

cust_ret

cust_ret['Retention_rate'] =cust_ret['Customer_retention_count'].pct_change()

cust_ret
```

	Order_Month_year	customer_unique_id_x	customer_unique_id_y	Customer_retention_count	Retention_rate
0	2016-09	1	1	0	NaN
1	2016-10	265	256	9	inf
2	2017-01	748	700	48	4.333333
3	2017-02	1641	1591	50	0.041667
4	2017-03	2546	2464	82	0.640000
5	2017-04	2303	2220	83	0.012195
6	2017-05	3545	3385	160	0.927711
7	2017-06	3135	2998	137	-0.143750
8	2017-07	3872	3718	154	0.124088
9	2017-08	4193	4011	182	0.181818
10	2017-09	4149	3958	191	0.049451
11	2017-10	4478	4302	176	-0.078534
12	2017-11	7288	7059	229	0.301136
13	2017-12	5513	5339	174	-0.240175
14	2018-01	7069	6834	235	0.350575
15	2018-02	6555	6296	259	0.102128
16	2018-03	7003	6805	198	-0.235521
17	2018-04	6798	6613	185	-0.065657
18	2018-05	6749	6558	191	0.032432
19	2018-06	6096	5942	154	-0.193717
20	2018-07	6156	5984	172	0.116883
21	2018-08	6351	6206	145	-0.156977

```
plt.figure(figsize=(15, 6))
sns.lineplot(data =cust_ret, x='Order_Month_year',y ='customer_unique_id_x',color='red',markers=True, dashes=True)
plt.xticks(rotation=90)

plt.twinx()
sns.barplot(data =cust_ret, x='Order_Month_year',y ='Retention_rate', color='lightblue')
plt.title('Customer Count and retention Rate')
plt.xlabel('Month-Year')
plt.ylabel('Customer Count', color='blue')
plt.ylabel('Retention Rate', color='red')
plt.tight_layout()
plt.show()
```

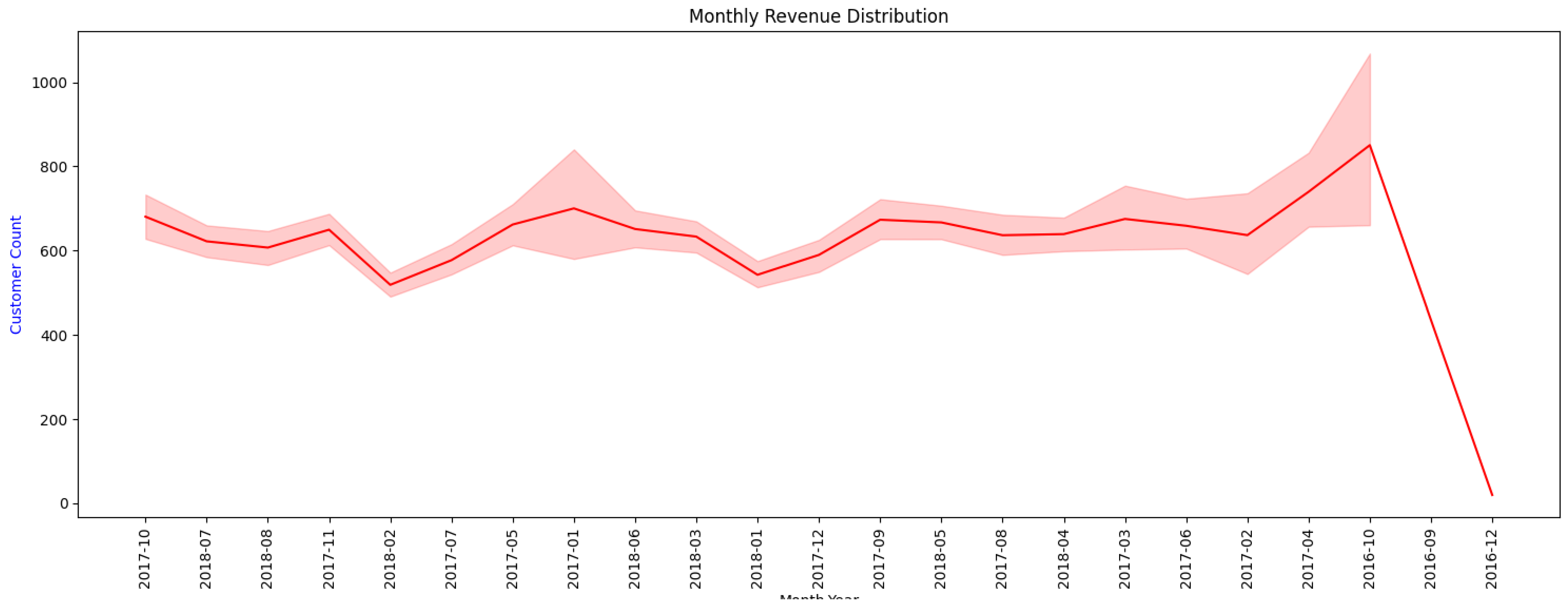


Insights from above chart:

- The line plot shows the trend in the total number of customers over time (represented by 'customer_unique_id_x'). Fluctuations in the customer count indicate changes in customer acquisition and churn rates across different months.
- The bar plot represents the retention rate for each month (represented by 'Retention_rate'). The retention rate indicates the proportion of customers from the previous month who continue to make purchases in the current month. Positive retention rates imply growth in the customer base, while negative retention rates suggest a decline.
- Observing the relationship between the total customer count and retention rate per month provides insights into customer loyalty and churn dynamics. Peaks or troughs in the retention rate may correlate with corresponding changes in the customer count.
- Patterns such as increases or decreases in both customer count and retention rate over consecutive months can highlight periods of significant customer growth or attrition.

How the revenues from existing/new customers on month on month basis

```
payment_order_data = pd.merge(orders_data,order_payment_data,how='left',on='order_id')

payment_order_data.head()

monthly_revenue = payment_order_data.groupby('Order_Month_year')['Revenue'].sum()

monthly_revenue

plt.figure(figsize=(15,6))
sns.lineplot(data =payment_order_data, x='Order_Month_year',y ='Revenue',color='red',markers=True, dashes=True)
plt.xticks(rotation=90)
plt.title('Monthly Revenue Distribution')
plt.xlabel('Month-Year')
plt.ylabel('Customer Count', color='blue')
plt.tight_layout()
plt.show()
```



Insights from above chart:

- The line plot illustrates the trend in monthly revenue over the specified period. Fluctuations in revenue levels across different months indicate variations in sales performance and business activity.
- Monitoring monthly revenue distribution allows businesses to evaluate their financial performance over time. Positive trends and consistent growth in revenue suggest a healthy business trajectory, while declines may indicate challenges or areas for improvement.

Understand the trends/seasonality of sales, quantity by category, location, month, week, day, time, channel, payment method etc...

1.Category wise Revenue

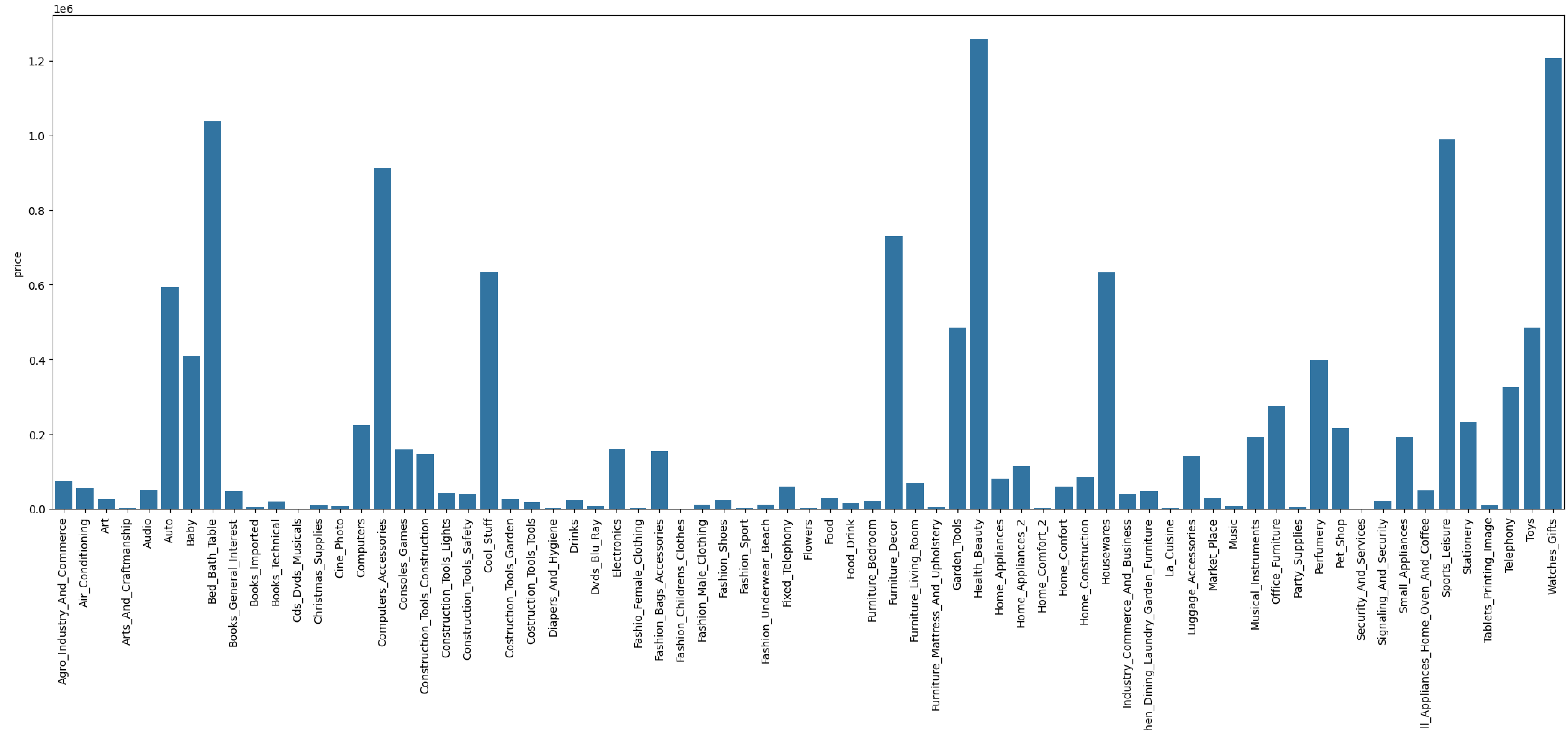
```
product_and_order_items_data = pd.merge(left=product_data,right=order_items_data,on='product_id')

Category_Wise_Reveue = product_and_order_items_data.groupby('product_category_name')['price'].sum().reset_index()

Category_Wise_Reveue
```

	product_category_name	price
0	Agro_Industry_And_Commerce	72530.47
1	Air_Conditioning	55024.96
2	Art	24202.64
3	Arts_And_Craftmanship	1814.01
4	Audio	50688.50
5	Auto	592720.11
6	Baby	409830.89
7	Bed_Bath_Table	1036988.68
8	Books_General_Interest	46856.88
9	Books_Imported	4639.85
10	Books_Technical	19096.06
11	Cds_Dvds_Musicals	730.00
12	Christmas_Supplies	8800.82
13	Cine_Photo	6933.46
14	Computers	222963.13
15	Computers_Accessories	911954.32
16	Consoles_Games	157465.22
17	Construction_Tools_Construction	144677.59
18	Construction_Tools_Lights	41080.00
19	Construction_Tools_Safety	40544.52
20	Cool_Stuff	635290.85
21	Costruction_Tools_Garden	25715.89
22	Costruction_Tools_Tools	15903.95
23	Diapers_And_Hygiene	1567.59
24	Drinks	22428.70
25	Dvds_Blu_Ray	5999.39
26	Electronics	160246.74
27	Fashio_Female_Clothing	2803.64
28	Fashion_Bags_Accessories	152823.54
29	Fashion_Childrens_Clothes	569.85
30	Fashion_Male_Clothing	10797.82
31	Fashion_Shoes	23562.77
32	Fashion_Sport	2119.51
33	Fashion_Underwear_Beach	9541.55
34	Fixed_Telephony	59583.00
35	Flowers	1110.04
36	Food	29393.41
37	Food_Drink	15179.48
38	Furniture_Bedroom	20028.78
39	Furniture_Decor	729762.49
40	Furniture_Living_Room	68916.56
41	Furniture_Mattress_And_Upholstery	4368.08
42	Garden_Tools	485256.46
43	Health_Beauty	1258681.34
44	Home_Appliances	80171.53
45	Home_Appliances_2	113317.74
46	Home_Comfort_2	760.27
47	Home_Confort	58572.04
48	Home_Construction	83088.12
49	Housewares	632248.66
50	Industry_Commerce_And_Business	39669.61
51	Kitchen_Dining_Laundry_Garden_Furniture	46328.37
52	La_Cuisine	2054.99
53	Luggage_Accessories	140429.98
54	Market_Place	28378.47
55	Music	6034.35
56	Musical_Instruments	191498.88
57	Office_Furniture	273960.70
58	Party_Supplies	4485.18
59	Perfumery	399124.87
60	Pet_Shop	214315.41
61	Security_And_Services	283.29
62	Signaling_And_Security	21509.23
63	Small_Appliances	190648.58
64	Small_Appliances_Home_Oven_And_Coffee	47445.71
65	Sports_Leisure	988048.97
66	Stationery	230943.23
67	Tablets_Printing_Image	7528.41
68	Telephony	323667.53
69	Toys	483946.60
[illegible]	[illegible]	[illegible]

70 Watches_Gifts 1205005.68

Next steps: Generate code with Category_Wise_Reveue View recommended plots

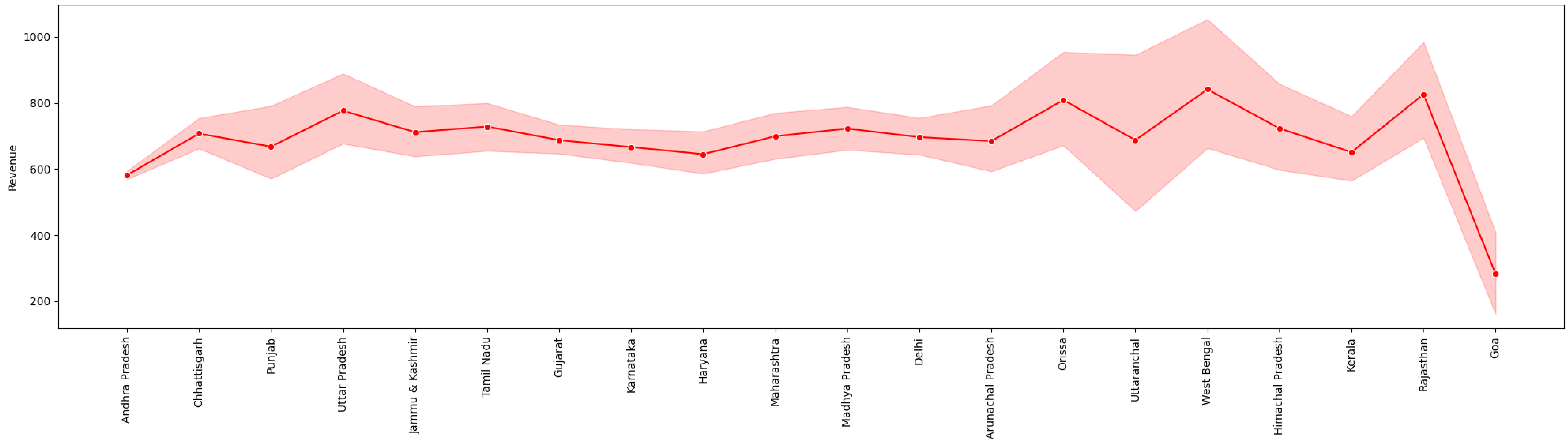
```
plt.figure(figsize=(20,10))
sns.barplot(x='product_category_name',y='price',data=Category_Wise_Reveue)
plt.xticks(rotation=90)
plt.xlabel('Month-Year')
plt.tight_layout()
plt.show()
```



Revenue By Location

```
customer_payment_location_data = pd.merge(payment_order_data,customers_data,on='customer_id')

customer_payment_location_data.columns

location_revenue = customer_payment_location_data.groupby('customer_state')['Revenue'].sum()

print(location_revenue.reset_index())

plt.figure(figsize=(20,6))
sns.lineplot(x='customer_state',y='Revenue',data=customer_payment_location_data,marker='o',color='red')
plt.xticks(rotation=90)
plt.xlabel('States')
plt.ylabel('Revenue')
plt.tight_layout()
plt.show()
```

	customer_state	Revenue
0	Andhra Pradesh	35762827.39
1	Arunachal Pradesh	776655.41
2	Chhattisgarh	4553290.58
3	Delhi	2355077.29
4	Goa	2825.88
5	Gujarat	5187133.03
6	Haryana	1451275.52
7	Himachal Pradesh	531237.05
8	Jammu & Kashmir	1101070.05
9	Karnataka	2600066.13
10	Kerala	1007363.44
11	Madhya Pradesh	2087130.58
12	Maharashtra	1409997.03
13	Orissa	511864.36
14	Punjab	458609.46
15	Rajasthan	676481.42
16	Tamil Nadu	1570281.01
17	Uttar Pradesh	808915.90
18	Uttaranchal	69415.49
19	West Bengal	307725.11



Insights from above chart:

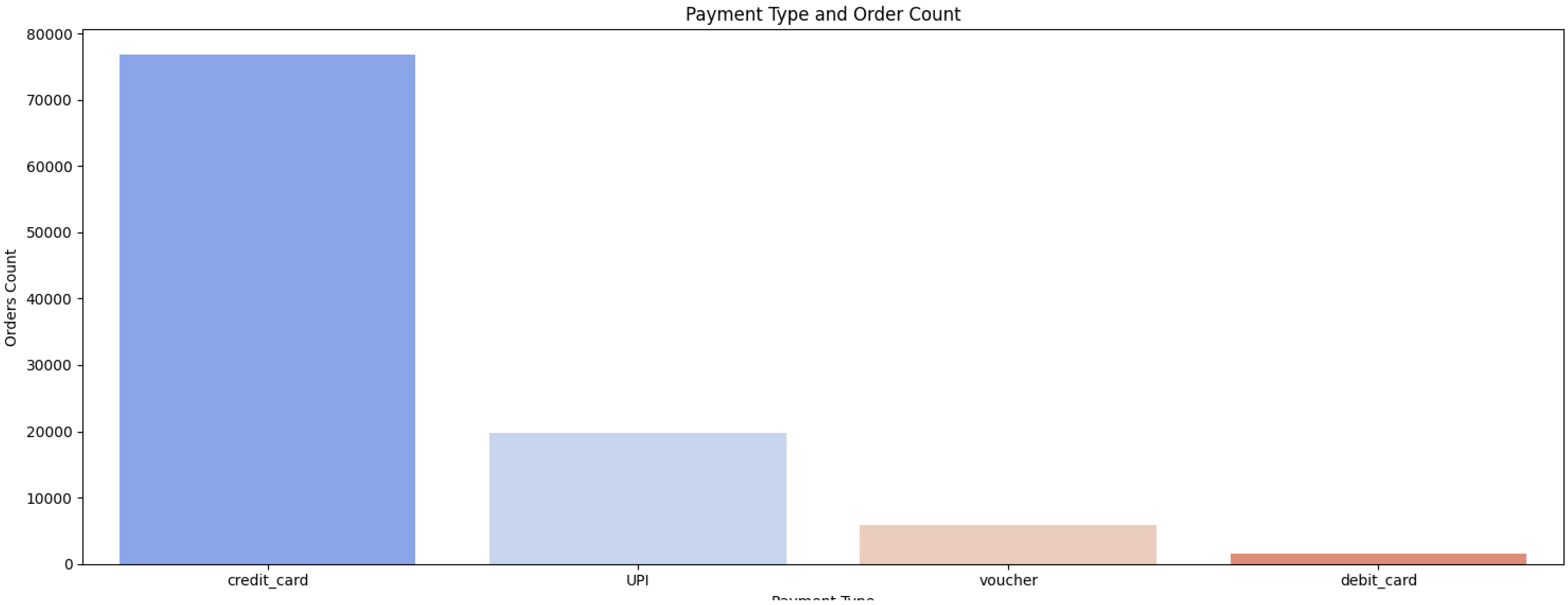
- The line plot provides a visual representation of revenue distribution across different states. It allows for a quick comparison of revenue contributions from various regions.

- States like Andhra Pradesh, Gujarat, and Chhattisgarh appear to contribute significantly to the total revenue. On the other hand, states like Goa, Uttaranchal, and West Bengal have relatively lower revenue contributions.

Trend of sales & quantity by Channel/Payment Method

```
order_payment_data = order_payment_data[order_payment_data['payment_type']!='not_defined']

payment_method_trend = order_payment_data.groupby('payment_type').agg({'order_id':'count','payment_value':'sum'}).reset_index()

print(payment_method_trend)

plt.figure(figsize=(15, 6))
sns.countplot(x='payment_type',data=order_payment_data,palette='coolwarm')

plt.title('Payment Type and Order Count')
plt.xlabel('Payment Type')
plt.ylabel('Orders Count')
plt.tight_layout()
plt.show()
```

```
  payment_type  order_id  payment_value
0          UPI     19784     2869361.27
1  credit_card     76795    12542084.19
2   debit_card      1529      217989.79
3      voucher      5775      379436.87
<ipython-input-253-71177e5b75f9>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='payment_type',data=order_payment_data,palette='coolwarm')
```



Insights from above plot:

- The countplot illustrates the distribution of orders across different payment types. It provides insight into the popularity and usage of various payment methods among customers.
- The plot indicates that the most commonly used payment methods are credit card and UPI (Unified Payments Interface), as they have significantly higher order counts compared to debit card and voucher payments.
- Understanding payment preferences can help businesses tailor their payment processing infrastructure to accommodate the preferred methods of their customers.
- Identifying less popular payment methods, such as debit card and voucher payments, may present opportunities for targeted promotional campaigns or incentives to encourage their adoption.

Popular Products by month, seller, state, category.

```
populer_product = product_data.groupby('product_category_name')['product_id'].count()

populer_product_sorted = populer_product.sort_values(ascending=False).reset_index()

populer_product_sorted

plt.figure(figsize=(15,8))
sns.barplot(y='product_id',x='product_category_name',data=populer_product_sorted)
plt.xticks(rotation=90)
plt.xlabel('Product Count')
plt.ylabel('Product Category')
plt.tight_layout()
plt.show()
```