📘 Project Summary: Brazilian Sales Data Analysis using mysql and python.

📊 Objective:

To analyze Brazilian e-commerce sales data, discover customer behavior patterns, and identify opportunities for improving logistics and customer retention.

In [37]:
```
# Installation of the mysql-connector-python package
#!pip install mysql-connector-python
```

First we will dump the e-commerce dataset into **mysql database** using the following steps:

In [38]:
```
#import pandas as pd
#import mysql.connector
#import os

# List of CSV files and their corresponding table names downloaded from Kaggle p
#csv_files = [
    #('customers.csv', 'customers'),
    #('orders.csv', 'orders'),
    #('sellers.csv', 'sellers'),
    #('products.csv', 'products'),
    #('geolocation.csv', 'geolocation'),
    #('order_items.csv','order_items'),
    #('payments.csv', 'payments')
#]

# Connect to the MySQL database
#conn = mysql.connector.connect(
    #host='localhost',
    #user='root',
    #password='890',
    #database='ecommerce'
#)
#cursor = conn.cursor()

# Folder containing the CSV files
#folder_path = 'E:\\python projects\\e_commerce'

#def get_sql_type(dtype):
    #if pd.api.types.is_integer_dtype(dtype):
     #   return 'INT'
    #elif pd.api.types.is_float_dtype(dtype):
     #   return 'FLOAT'
    #elif pd.api.types.is_bool_dtype(dtype):
```

```
     #   return 'BOOLEAN'
    #elif pd.api.types.is_datetime64_any_dtype(dtype):
     #   return 'DATETIME'
    #else:
     #   return 'TEXT'

#for csv_file, table_name in csv_files:
    #file_path = os.path.join(folder_path, csv_file)

    # Read the CSV file into a pandas DataFrame
    #df = pd.read_csv(file_path)

    # Replace NaN with None to handle SQL NULL
    #df = df.where(pd.notnull(df), None)

    # Debugging: Check for NaN values
    #print(f"Processing {csv_file}")
    #print(f"NaN values before replacement:\n{df.isnull().sum()}\n")

    # Clean column names
    #df.columns = [col.replace(' ', '_').replace('-', '_').replace('.', '_') for

    # Generate the CREATE TABLE statement with appropriate data types
    #columns = ', '.join([f'`{col}` {get_sql_type(df[col].dtype)}' for col in df
    #create_table_query = f'CREATE TABLE IF NOT EXISTS `{table_name}` ({columns}
    #cursor.execute(create_table_query)

    # Insert DataFrame data into the MySQL table
    #for _, row in df.iterrows():
        # Convert row to tuple and handle NaN/None explicitly
      #  values = tuple(None if pd.isna(x) else x for x in row)
     #   sql = f"INSERT INTO `{table_name}` ({', '.join(['`' + col + '`' for col
       # cursor.execute(sql, values)

    # Commit the transaction for the current CSV file
    #conn.commit()

# Close the connection
#conn.close()
```

After succesfull conncetion to the mysql database, the analysis and visualisation with key findings are as follows.

In [39]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector

db=mysql.connector.connect(host="localhost",
                           username="root",
                           password="890",
                           database="ecommerce")
cur=db.cursor()
```

1. List of all the unique cities where customers are located.

In [40]:
```
query="""select distinct customer_city from customers"""
cur.execute(query)
data=cur.fetchall()
data
df=pd.DataFrame(data, columns=["City_name"])
df.head()
```

Out[40]:

	City_name
0	franca
1	sao bernardo do campo
2	sao paulo
3	mogi das cruzes
4	campinas

As we can observe that customer are from top 5 cities namely Franca, Sao bernardo do campo...etc.

2. Count the number of orders placed in 2017 and 2018:

In [41]:
```
query="""SELECT COUNT(order_id) FROM orders WHERE YEAR(order_purchase_timestamp)
cur.execute(query)
data=cur.fetchall()
data
"total order placed in 2017 are", data[0][0]
```

Out[41]:
```
('total order placed in 2017 are', 90202)
```

In [42]:
```
query="""SELECT COUNT(order_id) FROM orders WHERE YEAR(order_purchase_timestamp)
cur.execute(query)
data=cur.fetchall()
data
"total order placed in 2018 are", data[0][0]
```

Out[42]:
```
('total order placed in 2018 are', 108022)
```

📦 Interpretation of Orders Placed in 2017 and 2018

In 2017, a total of 90,202 orders were placed, while in 2018, the number of orders increased to 108,022. This indicates a significant growth in customer activity or business performance year over year. The rise in order volume suggests an expanding customer base, improved marketing strategies, or increased product availability. Overall, the platform experienced positive growth in sales volume from 2017 to 2018.

3. The total sales per product category is:

In [43]:
```
query="""SELECT upper(products.product_category) Product_category, round(sum(pay
FROM products join order_items on products.product_id=order_items.product_id
join payments on payments.order_id=order_items.order_id
group by Product_category order by sales desc """
cur.execute(query)
data=cur.fetchall()
data
df=pd.DataFrame(data, columns=["Product_Category","Sales"])
df
```

Out[43]:

	Product_Category	Sales
0	BED TABLE BATH	3425107.34
1	HEALTH BEAUTY	3314746.24
2	COMPUTER ACCESSORIES	3170660.89
3	FURNITURE DECORATION	2860352.78
4	WATCHES PRESENT	2858433.36
...	...	...
69	PC GAMER	4348.86
70	HOUSE COMFORT 2	3421.08
71	CDS MUSIC DVDS	2398.86
72	FASHION CHILDREN'S CLOTHING	1571.34
73	INSURANCE AND SERVICES	649.02

74 rows × 2 columns

📊 Interpretation of Sales by Product Category

The sales data reveals that the top-performing product categories are "BED TABLE BATH", "HEALTH BEAUTY", and "COMPUTER ACCESSORIES", each generating over 3 million in sales. These categories likely represent essential or frequently purchased items, indicating strong consumer demand.

Mid-level categories, while not shown in full here, probably include items with moderate popularity or seasonal demand. They may still contribute significantly to overall revenue but do not match the performance of the top few.

At the lower end, categories like "PC GAMER", "HOUSE COMFORT 2", "CDS MUSIC DVDS", "FASHION CHILDREN'S CLOTHING", and "INSURANCE AND SERVICES" have generated very low sales, with the lowest being under ₹1,000. These categories may be niche, outdated, or not well-promoted in the current sales environment.

Overall, the data shows a large gap between the best- and worst-selling product categories, suggesting that consumer preferences are concentrated around a few key areas.

4. Calculation of the % of orders that were paid in installements is:

In [44]:
```
query="""SELECT sum(case when payment_installments>=1 then 1 else 0 end)/count(*
cur.execute(query)
data=cur.fetchall()
data
"the % of orders that were paid in installments is", data[0][0]
```

Out[44]:
```
('the % of orders that were paid in installments is', Decimal('99.9981'))
```

💳 Interpretation of Payment Installments

The percentage of orders that were paid in installments is 99.9981%. This shows that nearly all customers chose to pay using installment plans rather than full upfront payments. It indicates a strong preference for flexible payment options, which might be influenced by the availability of installment offers, the higher value of products, or general customer behavior favoring deferred payments.

5. the number of customers from each state is:

In [45]:
```
query="""SELECT customer_state, count(customer_id) from customers group by custo
cur.execute(query)
data=cur.fetchall()
data

"the the number of customers from each state.", data[0][0]
df=pd.DataFrame(data, columns=["state","customer_count"])
df
```

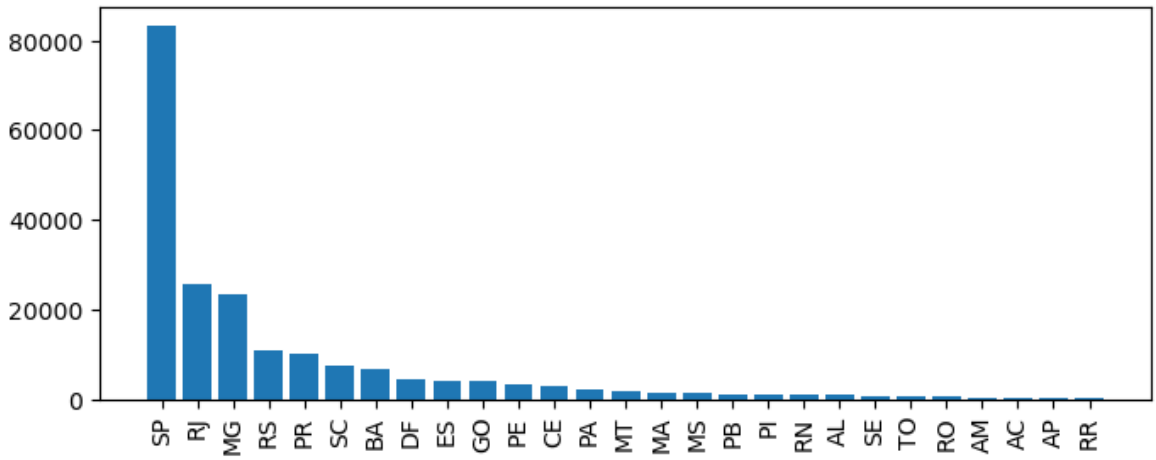
Out[45]:

	state	customer_count
0	SP	83492
1	SC	7274
2	MG	23270
3	PR	10090
4	RJ	25704
5	RS	10932
6	PA	1950
7	GO	4040
8	ES	4066
9	BA	6760
10	MA	1494
11	MS	1430
12	CE	2672
13	DF	4280
14	RN	970
15	PE	3304
16	MT	1814
17	AM	296
18	AP	136
19	AL	826
20	RO	506
21	PB	1072
22	TO	560
23	PI	990
24	AC	162
25	SE	700
26	RR	92

Visualisation of total number of customer's across different cities using BAR chart:

In [46]:
```
query="""SELECT customer_state, count(customer_id) from customers group by custo
cur.execute(query)
data=cur.fetchall()
data

"the the number of customers from each state.", data[0][0]
df=pd.DataFrame(data, columns=["state","customer_count"])
df
df=df.sort_values(by="customer_count", ascending=False)
plt.figure(figsize=(8,3))
plt.bar(df["state"], df["customer_count"])
plt.xticks(rotation=90)
plt.show()
```



🌎 Interpretation of Customer Distribution by State

The state with the highest number of customers is SP (São Paulo), with 83,492 customers, followed by RJ (Rio de Janeiro) and MG (Minas Gerais), with 25,704 and 23,270 customers respectively. These states are among the most populous and economically active regions, which likely contributes to their higher customer counts.

On the other hand, states like RR (Roraima), AP (Amapá), and AC (Acre) have the lowest number of customers, each with fewer than 200. These are smaller or less densely populated regions, which may explain the lower customer engagement or market penetration.

Overall, the customer base is heavily concentrated in the southeastern and southern parts of the country, reflecting regional disparities in market activity, infrastructure, or access to services.

6. The total number of orders per month in year 2018 is:

In [47]:
```
query="""SELECT monthname(order_purchase_timestamp) months, count(order_id) orde
from orders where year(order_purchase_timestamp)
group by months"""
cur.execute(query)
data=cur.fetchall()
```

```
data
df=pd.DataFrame(data, columns=["Months", "order_count"])
```

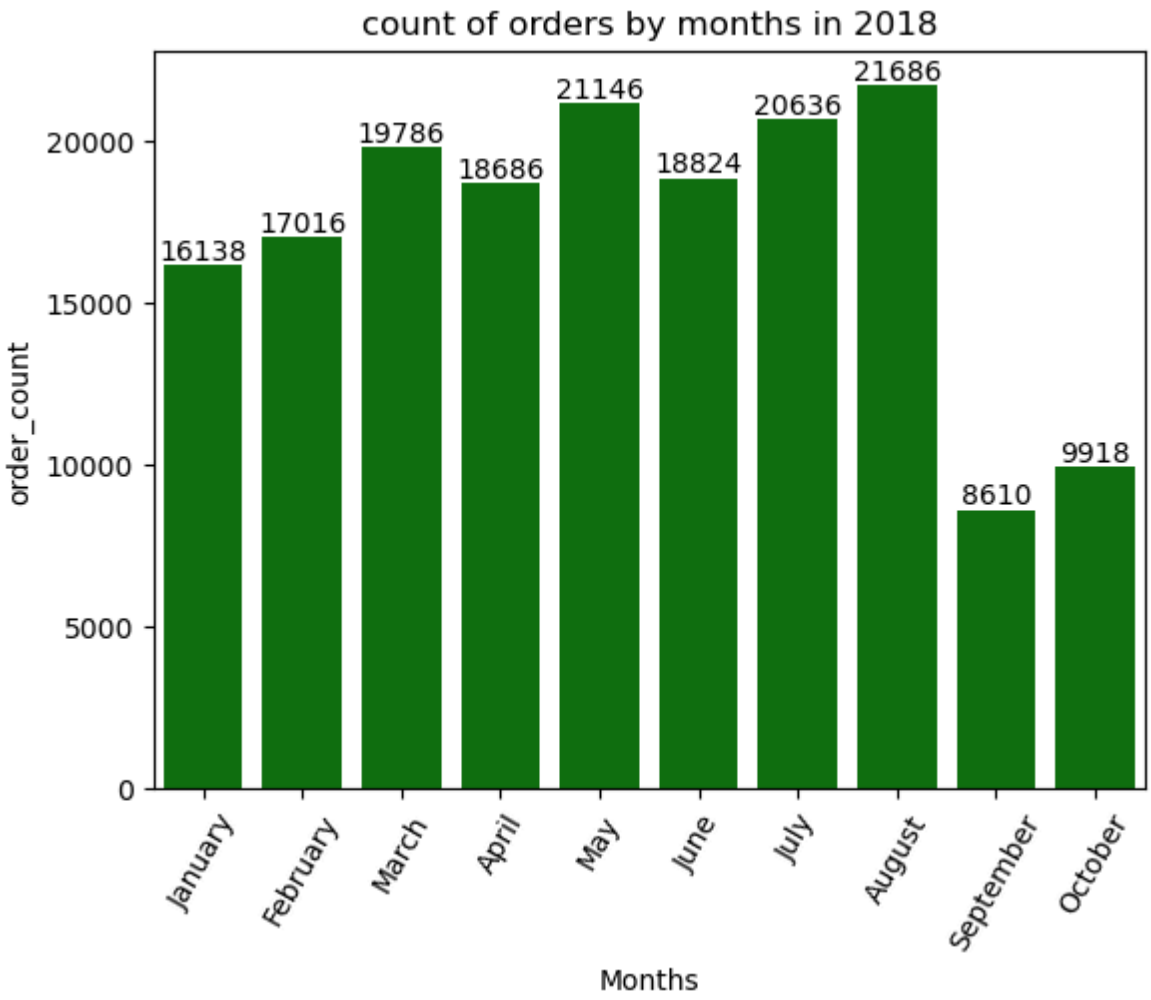
In [48]:
```
df=pd.DataFrame(data, columns=["Months", "order_count"])
df
```

Out[48]:

	Months	order_count
0	October	9918
1	July	20636
2	August	21686
3	November	15088
4	February	17016
5	April	18686
6	May	21146
7	January	16138
8	June	18824
9	March	19786
10	December	11348
11	September	8610

Visualisation of total number of orders per months in the year 2018:

In [49]:
```
o=["January", "February", "March","April","May","June","July", "August","Septemb
ax=sns.barplot(x=df["Months"],y=df["order_count"], data=df, order=o, color="gree
ax.bar_label(ax.containers[0])
plt.xticks(rotation=60)
plt.title("count of orders by months in 2018")
plt.show()
```



Interpretation of Monthly Order Distribution

The month with the highest number of orders is August, with 21,686 orders, closely followed by May and July. This suggests that mid-year months tend to have higher customer activity, possibly due to seasonal trends, promotions, or holidays during that period.

On the other hand, months like September, October, and December show relatively lower order volumes, with October and September having the lowest counts at 9,918 and 8,610 respectively. This may indicate a slower shopping period or reduced campaign activity during these months.

Overall, order volume fluctuates significantly throughout the year, indicating that customer purchasing behavior is influenced by time-based factors such as sales events, seasonal demands, or economic cycles.

7. The average numbers of products per order, grouped by customer city is:

In [50]:
```
query="""with count_per_order as (select orders.order_id, orders.customer_id, co
from orders join order_items on orders.order_id= order_items.order_id
group by orders.order_id, orders.customer_id)
```

```
select customers.customer_city, round(avg(count_per_order.oc),2) avg_order
from customers join count_per_order on customers.customer_id=count_per_order.cus
group by customers.customer_city order by avg_order desc"""
cur.execute(query)
data=cur.fetchall()
data
df=pd.DataFrame(data, columns=["customer_city", "avg_products_per_order"])

df.head(10)
```

Out[50]:

	customer_city	avg_products_per_order
0	padre carvalho	14.00
1	celso ramos	13.00
2	datas	12.00
3	candido godoi	12.00
4	matias olimpio	10.00
5	cidelandia	8.00
6	curralinho	8.00
7	picarra	8.00
8	morro de sao paulo	8.00
9	teixeira soares	8.00

🏙️ Interpretation of Average Products per Order by City

The data shows that cities like Padre Carvalho, Celso Ramos, and Datas have the highest average number of products per order, with Padre Carvalho leading at 14 products per order. These unusually high averages may indicate bulk buying behavior, possibly due to limited access to local retail or a tendency to consolidate purchases in fewer orders.

Cities like Cidelandia, Curralinho, and Teixeira Soares also show higher-than-average product counts per order, averaging around 8 products. This suggests a pattern where customers in smaller or rural cities may prefer to place fewer orders with more items per transaction.

Overall, customer behavior varies by location, and cities with higher averages may represent opportunities for offering volume-based discounts or promotions on large orders.

8. The % of total revenue contributed by each product category is:

In [51]:

```
query="""SELECT upper(products.product_category) category,
round((sum(payments.payment_value)/(select sum(payment_value) from payments))*10
FROM products join order_items on products.product_id=order_items.product_id
```

```
join payments on payments.order_id=order_items.order_id
group by category order by sales_percentage desc"""
cur.execute(query)
data=cur.fetchall()
data
df=pd.DataFrame(data, columns=["category", "percentage_sales"])

df.head(10)
```

Out[51]:

	category	percentage_sales
0	BED TABLE BATH	21.40
1	HEALTH BEAUTY	20.71
2	COMPUTER ACCESSORIES	19.81
3	FURNITURE DECORATION	17.87
4	WATCHES PRESENT	17.86
5	SPORT LEISURE	17.39
6	HOUSEWARES	13.68
7	AUTOMOTIVE	10.65
8	GARDEN TOOLS	10.47
9	COOL STUFF	9.74

💰 Interpretation of Revenue Contribution by Product Category

The category contributing the highest percentage of total revenue is "BED TABLE BATH", accounting for 21.40% of total sales, followed closely by "HEALTH BEAUTY" and "COMPUTER ACCESSORIES", contributing 20.71% and 19.81% respectively. These categories clearly dominate revenue generation, likely due to their broad utility, frequent demand, or high average order value.

Other significant contributors include "FURNITURE DECORATION", "WATCHES PRESENT", and "SPORT LEISURE", each contributing between 17% and 18%. These categories may represent popular lifestyle or gifting segments.

Overall, revenue is concentrated among a few top-performing product categories, which reflects consumer preferences and possibly the effectiveness of marketing, pricing, and inventory strategies in those areas.

9. The correlation between product price and the number of times a product has been purchased is:

In [52]:
```
query="""select products.product_category, count(order_items.product_id),
round(avg(order_items.price),2)
```

```
from products join order_items on
products.product_id=order_items.product_id
group by products.product_category"""
cur.execute(query)
data=cur.fetchall()
data
df=pd.DataFrame(data, columns=["category", "order_count", "price"])

df.head(10)
```

Out[52]:

	category	order_count	price
0	HEALTH BEAUTY	19340	130.16
1	sport leisure	17282	114.34
2	Cool Stuff	7592	167.36
3	computer accessories	15654	116.51
4	Watches present	11982	201.14
5	housewares	13928	90.79
6	electronics	5534	57.91
7	None	3206	112.00
8	toys	8234	117.55
9	bed table bath	22230	93.30

In [53]:
```
import numpy as np
arr1=df["order_count"]
arr2=df["price"]
np.corrcoef([arr1, arr2])
```

Out[53]:
```
array([[ 1.        , -0.10631514],
       [-0.10631514,  1.        ]])
```

📈 Interpretation of Correlation Between Product Price and Purchase Count

From the data, there doesn't appear to be a strong positive or negative correlation between the average price of a product category and the number of times it has been purchased. For instance, "BED TABLE BATH" has the highest order count (22,230) despite a relatively moderate average price of ₹93.30, while "COOL STUFF" has fewer orders (7,592) but a higher average price of ₹167.36.

Categories like "HEALTH BEAUTY" and "SPORT LEISURE" also show high purchase counts with mid-range prices, while "WATCHES PRESENT" has a higher price (₹201.14) and fewer orders in comparison. This suggests that purchase frequency is not necessarily driven by price alone, but likely influenced by product utility, demand, customer preferences, and category relevance.

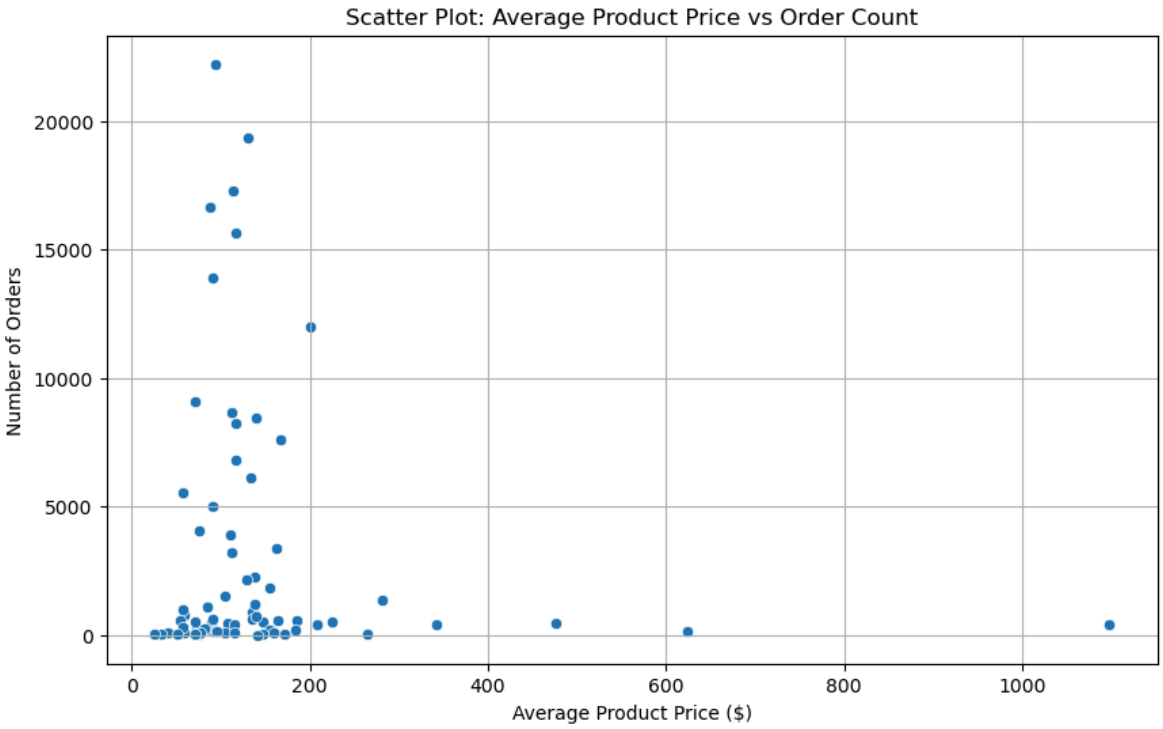
Overall, the relationship between price and purchase volume appears weak or inconsistent across categories, indicating that other factors are playing a more significant

role in influencing buying behavior.

scatter plot of order count vs. average product price:

In [54]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,6))
sns.scatterplot(x='price', y='order_count', data=df)
plt.title('Scatter Plot: Average Product Price vs Order Count')
plt.xlabel('Average Product Price ($)')
plt.ylabel('Number of Orders')
plt.grid(True)
plt.show()
```



📉 Interpretation of Scatter Plot: Average Product Price vs. Order Count

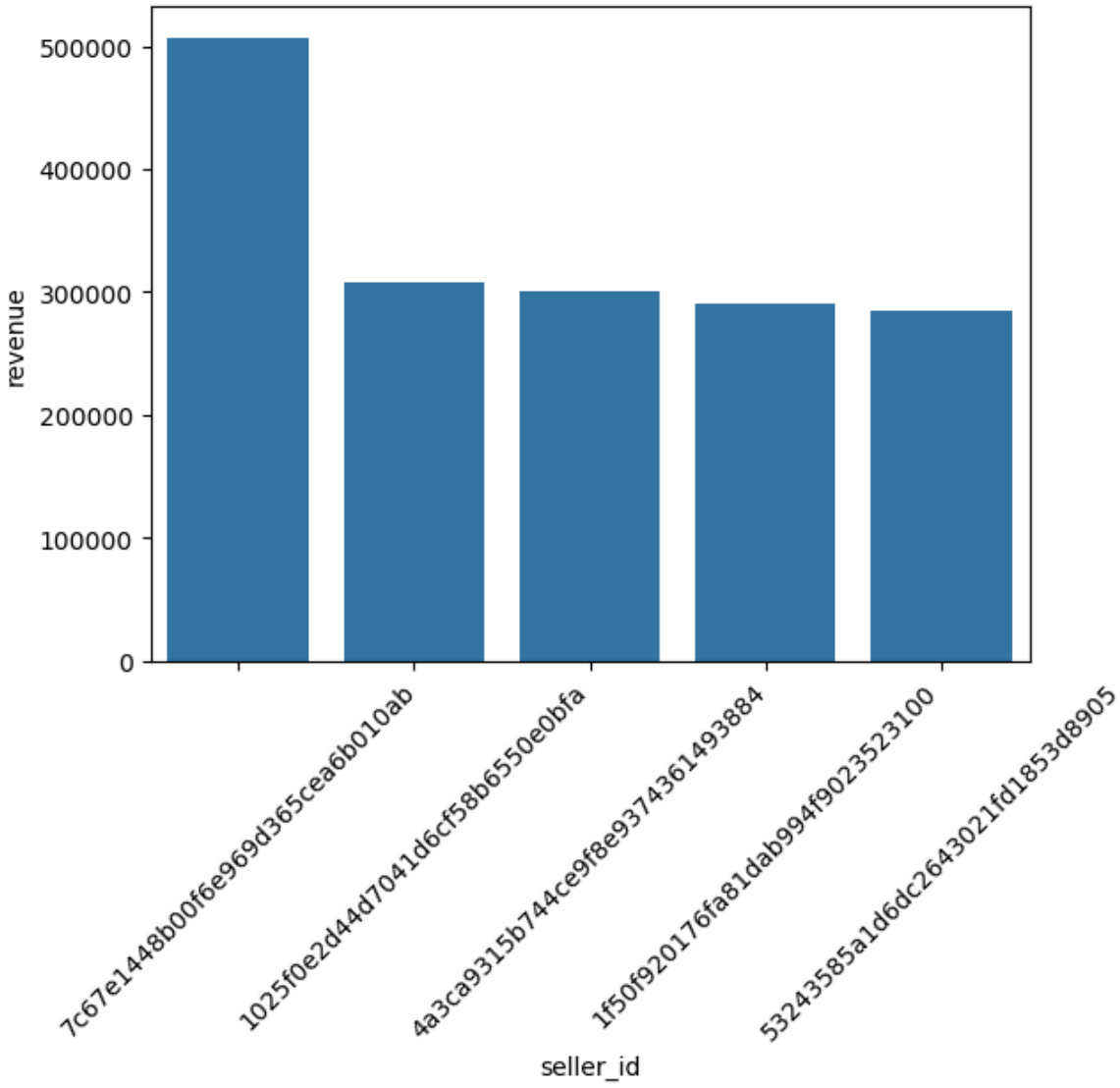
The scatter plot shows the relationship between the average price of products in each category and the number of times they were ordered. The data points appear widely scattered without forming a clear upward or downward trend, indicating the absence of a strong linear relationship.

Some categories with relatively low prices have very high order counts, such as "BED TABLE BATH", while others with higher average prices, like "WATCHES PRESENT" or "COOL STUFF", have comparatively fewer orders. This suggests that products with lower prices tend to be ordered more frequently, but the pattern is not strong or consistent across all categories.

Overall, the plot supports the earlier statistical result showing a weak negative correlation. Price does not appear to be a dominant factor in determining how frequently a product category is purchased, and other factors like product type, necessity, and popularity are likely influencing customer behavior.

10. The total revenue generated by each seller, and ranked by revenue is:

In [55]:
```
query="""select*, dense_rank() over(order by revenue desc) as rn from
(select order_items.seller_id, sum(payments.payment_value) revenue
from order_items join payments on order_items.order_id=payments.order_id
group by order_items.seller_id) as a"""
cur.execute(query)
data=cur.fetchall()
data
df=pd.DataFrame(data, columns=["seller_id", "revenue", "ranks"])
df=df.head()
sns.barplot(x="seller_id", y="revenue", data=df)
plt.xticks(rotation=45)
plt.show()
```



In [56]:
```
df=df.head()
df
```

Out[56]:

	seller_id	revenue	ranks
0	7c67e1448b00f6e969d365cea6b010ab	507166.907302	1
1	1025f0e2d44d7041d6cf58b6550e0bfa	308222.039840	2
2	4a3ca9315b744ce9f8e9374361493884	301245.269765	3
3	1f50f920176fa81dab994f9023523100	290253.420128	4
4	53243585a1d6dc2643021fd1853d8905	284903.080498	5

💼 Interpretation of Top 5 Sellers by Revenue:

The table and bar plot show the top 5 sellers ranked by total revenue in dollars. The leading seller (`7c67e1448b00f6e969d365cea6b010ab`) generated about $507,167, which is significantly higher than the second-ranked seller with approximately $308,222 in revenue.

There is a clear gap between the top seller and the others, indicating that the highest-ranked seller dominates in terms of sales. Sellers ranked 2 to 5 have closer revenue values, ranging between roughly $285,000 and $308,000, suggesting strong competition within this group.

These sellers likely excel due to high product demand, competitive pricing, or efficient operations, making them key contributors to the platform's overall revenue.

11. The Moving average of order values for each customers over their order history is:

In [57]:
```
query="""select customer_id, order_purchase_timestamp, payment,
avg(payment) over (partition by customer_id order by order_purchase_timestamp
rows between 2 preceding and current row) as mov_avg
from
(select orders.customer_id, orders.order_purchase_timestamp,
payments.payment_value as payment
from payments join orders on payments.order_id=orders.order_id) as a;
"""
cur.execute(query)
data=cur.fetchall()
df=pd.DataFrame(data, columns=["customer_id", "timestamp", "price", "moving_aver
df
```

Out[57]:

	customer_id	timestamp	price	moving_average
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
1	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
2	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
3	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
4	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.42	195.419998
...	...	...	...	...
207767	ffff42319e9b2d713724ae527742af25	2018-06-13 16:57:05	214.13	214.130005
207768	ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.500000
207769	ffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.500000
207770	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001
207771	ffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001

207772 rows × 4 columns

📊 Interpretation of Moving Average of Order Values per Customer

The moving average column represents the average payment amount for each customer calculated over their current order and the two previous orders, ordered by purchase date. This smooths out short-term fluctuations in individual order values, helping to identify trends in a customer's purchasing behavior over time.

For example, if a customer's order values are generally increasing, their moving average will gradually rise, showing increased spending. Conversely, if order values drop or vary greatly, the moving average will smooth those changes to provide a clearer view of the overall trend.

This metric is useful for understanding customer loyalty and purchasing patterns, as it highlights whether customers tend to increase, decrease, or maintain their spending across multiple purchases.

12. The cumulative sales per month for each year is:

In [58]:

```
query=""" select years, months, payment, sum(payment)
over (order by years, months) cumulative_sales from
(select year(orders.order_purchase_timestamp) as years,
month(orders.order_purchase_timestamp) as months,
round(sum(payments.payment_value),2) as payment
from orders join payments on orders.order_id=payments.order_id
group by years, months order by years, months) as a;

"""
cur.execute(query)
data=cur.fetchall()
df=pd.DataFrame(data, columns=["years", "months", "payment", "cumulative_sales"]
df
```

Out[58]:

	years	months	payment	cumulative_sales
0	2016	9	504.48	504.48
1	2016	10	118180.96	118685.44
2	2016	12	39.24	118724.68
3	2017	1	276976.08	395700.76
4	2017	2	583816.02	979516.78
5	2017	3	899727.20	1879243.98
6	2017	4	835576.06	2714820.04
7	2017	5	1185837.64	3900657.68
8	2017	6	1022552.76	4923210.44
9	2017	7	1184765.84	6107976.28
10	2017	8	1348792.64	7456768.92
11	2017	9	1455524.90	8912293.82
12	2017	10	1559355.76	10471649.58
13	2017	11	2389765.60	12861415.18
14	2017	12	1756802.96	14618218.14
15	2018	1	2230008.36	16848226.50
16	2018	2	1984926.68	18833153.18
17	2018	3	2319304.24	21152457.42
18	2018	4	2321570.96	23474028.38
19	2018	5	2307964.30	25781992.68
20	2018	6	2047761.00	27829753.68
21	2018	7	2133081.50	29962835.18
22	2018	8	2044850.64	32007685.82
23	2018	9	8879.08	32016564.90
24	2018	10	1179.34	32017744.24

📈 Interpretation of Cumulative Monthly Sales Over Years

The table shows the monthly sales and their cumulative totals from September 2016 through October 2018.

Starting with relatively low sales in late 2016, there is a rapid increase in monthly sales beginning in early 2017. Each month's sales are added to the previous months to give a running total of revenue generated over time.

The cumulative sales steadily grow, reflecting consistent and substantial revenue generation month after month. The sharp rise in cumulative sales through 2017 and 2018 suggests strong business growth and increasing customer purchases.

The small sales values in September and October 2018 compared to previous months might indicate incomplete data for those months or a drop in sales activity toward the end of the period.

Overall, this cumulative sales trend is a clear indicator of how total revenue accumulates over time, useful for tracking business performance and forecasting future growth.

The year-over-year growth rate of total sales is:

In [59]:
```
query="""with a as (select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as payment
from orders join payments on orders.order_id=payments.order_id
group by years order by years)
select years, ((payment-lag(payment, 1) over (order by years))/
lag(payment, 1) over (order by years))*100 from a

"""
cur.execute(query)
data=cur.fetchall()
df=pd.DataFrame(data, columns=["years", "y_o_y_growth"])
df
```

Out[59]:

	years	y_o_y_growth
0	2016	NaN
1	2017	12112.703761
2	2018	20.000924

📊 Interpretation of Year-over-Year Growth Rate of Total Sales

The year-over-year (YoY) growth rate measures how much the total sales have increased compared to the previous year.

- In 2016, there is no previous year to compare to, so the growth rate is not available.
- In 2017, there is a massive growth of approximately 12,112.7%, indicating an explosive increase in sales compared to 2016. This unusually high figure suggests that sales in 2016 were very low or the data for 2016 only covers part of the year.
- In 2018, sales continued to grow but at a much more moderate rate of 20%, indicating steady and healthy business expansion compared to 2017.

This YoY growth rate highlights the rapid scale-up phase of the business followed by a period of sustained growth.

14. The retention rate of customer, defined as the % of customers who make another purchase within 6 months of their first purchase is:

In [60]:
```
query="""WITH first_orders AS (
    SELECT
        customers.customer_id,
        MIN(orders.order_purchase_timestamp) AS first_order
    FROM customers
    JOIN orders ON customers.customer_id = orders.customer_id
    GROUP BY customers.customer_id
),
subsequent_orders AS (
    SELECT
        a.customer_id,
        COUNT(DISTINCT orders.order_id) AS purchase_count
    FROM first_orders a
    LEFT JOIN orders ON orders.customer_id = a.customer_id
        AND orders.order_purchase_timestamp > a.first_order
        AND orders.order_purchase_timestamp <= DATE_ADD(a.first_order, INTERVAL
    GROUP BY a.customer_id
)
SELECT
    COUNT(CASE WHEN purchase_count >= 1 THEN 1 END) * 100.0 / COUNT(*) AS retent
FROM subsequent_orders;"""

cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns=["repeated_purchase_withint_6_month(%)"])
df
```

Out[60]:

	repeated_purchase_withint_6_month(%)
0	0.00000

Customer Retention Interpretation

The retention rate measures the percentage of customers who make another purchase within 6 months of their first purchase.

If the retention rate is zero or extremely low, it indicates that customers generally do not return to make repeat purchases in that timeframe. This might signal issues such as lack of customer engagement, low satisfaction, or that the business model is focused on one-time purchases.

A zero retention rate highlights the need to investigate customer experience, loyalty programs, or marketing strategies to encourage repeat purchases.

14. The top 3 customers who spent the most money in each year is:

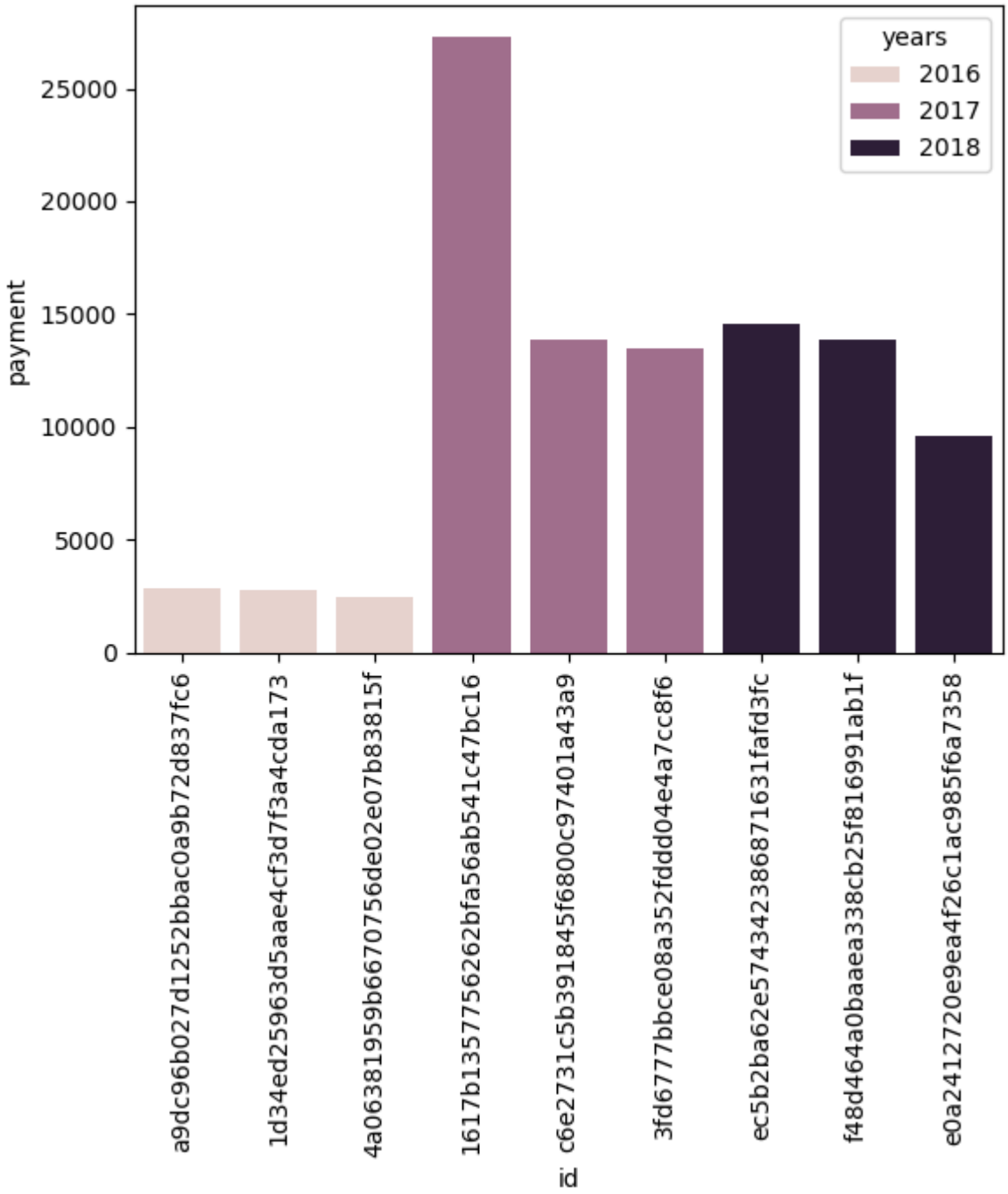
In [61]:
```
query="""select years, customer_id, payment, d_rank
from
(select year(orders.order_purchase_timestamp) years, orders.customer_id,
sum(payments.payment_value) payment,
dense_rank() over(partition by year(orders.order_purchase_timestamp)
order by sum(payments.payment_value) desc ) d_rank
from orders join payments
on payments.order_id=orders.order_id
group by year(orders.order_purchase_timestamp),
orders.customer_id) as a
where d_rank <=3;
"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns=["years","id","payment","rank"])
df
```

Out[61]:

	years	id	payment	rank
0	2016	a9dc96b027d1252bbac0a9b72d837fc6	2847.100098	1
1	2016	1d34ed25963d5aae4cf3d7f3a4cda173	2801.479980	2
2	2016	4a06381959b6670756de02e07b83815f	2455.560059	3
3	2017	1617b1357756262bfa56ab541c47bc16	27328.160156	1
4	2017	c6e2731c5b391845f6800c97401a43a9	13858.620117	2
5	2017	3fd6777bbce08a352fddd04e4a7cc8f6	13453.320312	3
6	2018	ec5b2ba62e574342386871631fafd3fc	14549.759766	1
7	2018	f48d464a0baaea338cb25f816991ab1f	13844.419922	2
8	2018	e0a2412720e9ea4f26c1ac985f6a7358	9618.879883	3

Visualisation of Top 3 customers who spents more money in each year:

In [62]:
```
sns.barplot(x="id", y="payment", data=df, hue="years")
plt.xticks(rotation=90)
plt.show()
```



Intrepretation of the Top 3 customers based on their spending in each year

The data presents the top three customers who spent the most money in each year from 2016 to 2018.

In 2016, the highest spending customer made purchases totaling approximately $2,847, followed closely by two others spending $2,801 and $2,455 respectively. These spending figures are relatively moderate.

In 2017, there is a significant increase in customer spending. The top customer spent over $27,328, while the second and third highest spent $13,858 and $13,453 respectively. This reflects a major growth in high-value purchases.

In 2018, the top spender contributed around $14,549 in total payments, followed by two others at $13,844 and $9,618. Although this is slightly lower than 2017's peak, it still shows higher spending than in 2016.

Overall, the data indicates that the top spenders in 2017 made unusually high-value purchases, while 2018 saw a slight drop but remained strong. This could point to promotions, loyalty behavior, or macroeconomic factors influencing customer purchase patterns across the years.

📊 Final Conclusion and Recommendations

📑 Conclusion

The data analysis of customer orders, payments, and product categories across Brazil's e-commerce platform reveals several important business insights:

- **Order Trends**: There was a sharp rise in total orders from 2017 (90,202) to 2018 (108,022), indicating strong growth in user activity and engagement.
- **Installment Payments**: Nearly all customers (≈ 99.99%) preferred to pay via installments, showing a strong dependence on flexible payment methods.
- **Geographical Insights**: The majority of customers came from the state of São Paulo (SP), followed by Minas Gerais (MG) and Rio de Janeiro (RJ), highlighting SP as the primary market.
- **Seasonal Ordering Patterns**: The highest order volumes occurred in August, May, and July, suggesting potential peaks in seasonal or promotional campaigns.
- **Product Preferences**: Certain small towns recorded high average products per order, while categories like *Bed Table Bath*, *Health Beauty*, and *Computer Accessories* contributed significantly to total revenue.
- **Price vs Popularity**: A slight negative correlation between product price and number of orders suggests that higher-priced items tend to be purchased less frequently.
- **Top Sellers**: A small group of sellers account for a large share of revenue, with the top seller generating over $507K, indicating potential concentration risk.
- **Customer Order Value Trends**: Moving averages show that some customers maintain consistent spending over multiple purchases.
- **Sales Momentum**: Monthly cumulative and YoY sales indicate continuous growth, with a remarkable 12112% growth from 2016 to 2017 due to low initial sales.
- **Customer Retention**: The retention rate within 6 months appears low or unidentifiable, suggesting customers may not return frequently.
- **Top Customers**: A few loyal customers drive substantial revenue yearly, especially in 2017.

✅ Recommendations

1. **Enhance Loyalty Programs**: Introduce targeted loyalty incentives or reward systems to improve customer retention and encourage repeat purchases, especially within

the first 6 months.

2. **Installment Optimization**: Given the popularity of installment payments, consider offering optimized financing plans or partnering with credit providers to make installment purchases even more attractive.

3. **Focus on High-Revenue States**: Prioritize marketing campaigns and logistics improvements in high-performing states like SP, MG, and RJ to maximize reach and ROI.

4. **Category Expansion**: Since a few product categories dominate revenue, consider diversifying within these segments (e.g., more premium Health & Beauty or Bed & Bath lines).

5. **Seller Diversification**: Encourage onboarding of more sellers and reduce over-dependence on top-performing sellers to minimize operational risk.

6. **Seasonal Campaigns**: Capitalize on order surges during May–August through discounts, bundle offers, or free shipping to drive even greater volume.

7. **Monitor Price Sensitivity**: Since there's a slight negative correlation between price and order volume, continue to balance pricing strategies—consider offering value bundles for higher-priced items.

8. **Customer Segmentation**: Identify and nurture high-value customers using CRM tools and exclusive offers to retain them.

By implementing these recommendations, the platform can improve customer experience, optimize revenue growth, and ensure long-term sustainability.