In [1]:
```
import matplotlib.pyplot as plt
```

In [2]:
```
import pandas as pd
```

In [3]:
```
%load_ext sql
```

In [ ]:
```
%env OPENAI_API_KEY=sk-proj-sub1S4R0W2Q6udeHSs4q0k5tb39EmAujo1Uo5HoFjeXgCHVhl1QpbTN
```

In [7]:
```
%%capture
%sql mysql+pymysql://root:Ak47&m4a1@localhost/db_netease
```

A basic view of data structure

In [8]:
```
%sql SELECT * FROM dwd_netease_order_detail limit 5;
```

```
 * mysql+pymysql://root:***@localhost/db_netease
5 rows affected.
```

Out[8]:

product_id	seller_id	
00066f42aeeb9f3007548bb9d3f33c38	5670f4db5b62c43d542e1b2d56b0cf7c	f30149f4a8882a0889
00088930e925c41fd95ebfe695fd2655	7142540dd4c91e2237acb7e911c4eba2	f5eda0ded77c1293b0
0009406fd7479715e4bef61dd91f2462	4a3ca9315b744ce9f8e9374361493884	0bf736fd0fd5169d6
000b8f95fcb9e0096488278317764d19	40ec8ab6cdafbcc4f544da38c67da39a	6f0dfb5b5398b271cc
000b8f95fcb9e0096488278317764d19	40ec8ab6cdafbcc4f544da38c67da39a	3aba44d8e554ab4bb

In [ ]:
```
```

In [9]:
```
%%sql
DROP VIEW IF EXISTS o_time;
CREATE VIEW o_time AS
SELECT
    order_id,
    customer_unique_id,
    YEAR(order_purchase_timestamp) AS y,
    MONTH(order_purchase_timestamp) AS m,
    DATE(order_purchase_timestamp) AS d,
```

```
    HOUR(order_purchase_timestamp) AS h
FROM dwd_netease_order_detail;
```

```
 * mysql+pymysql://root:***@localhost/db_netease
0 rows affected.
0 rows affected.
```

Out[9]: []

In [10]:
```
%%sql basic_inf <<
SELECT
    SUM(payment_value) AS Total_Payment,
    COUNT(order_id) AS Total_Order,
    SUM(payment_value) / COUNT(DISTINCT customer_unique_id) AS Price_per_customer,
    COUNT(DISTINCT customer_unique_id) AS Customer_Numbers,
    COUNT(DISTINCT seller_id) AS Seller_numbers,
    COUNT(DISTINCT product_category_name) AS SPU,
    COUNT(DISTINCT product_id) AS SKU
FROM dwd_netease_order_detail;
```

```
 * mysql+pymysql://root:***@localhost/db_netease
1 rows affected.
Returning data to local variable basic_inf
```

In [11]:
```
df_basic_inf = basic_inf.DataFrame()
df_basic_inf.round(2)
```

Out[11]:

	Total_Payment	Total_Order	Price_per_customer	Customer_Numbers	Seller_numbers	SP
0	14981309.77	95282	162.45	92221	2929	7

1.2 GMV Analysis

In [12]:
```
%%sql
yearly_GMV <<
SELECT
y Year,sum(payment_value) Yearly_GMV from dwd_netease_order_detail a1
LEFT JOIN o_time a2 on a1.customer_unique_id = a2.customer_unique_id
group by y
ORDER BY y;
```

```
 * mysql+pymysql://root:***@localhost/db_netease
2 rows affected.
Returning data to local variable yearly_GMV
```

In [13]:
```
%%sql
monthly_GMV <<
SELECT
case
when m<10 then concat(y,'-0',m)
when m>=10 then concat(y,'-',m)
end as Y_m,
sum(payment_value) YGMV from dwd_netease_order_detail b1
```

```
LEFT JOIN o_time b2 on b1.customer_unique_id = b2.customer_unique_id
GROUP BY case
when m<10 then concat(y,'-0',m)
when m>=10 then concat(y,'-',m)
end
ORDER BY Y_m;
```

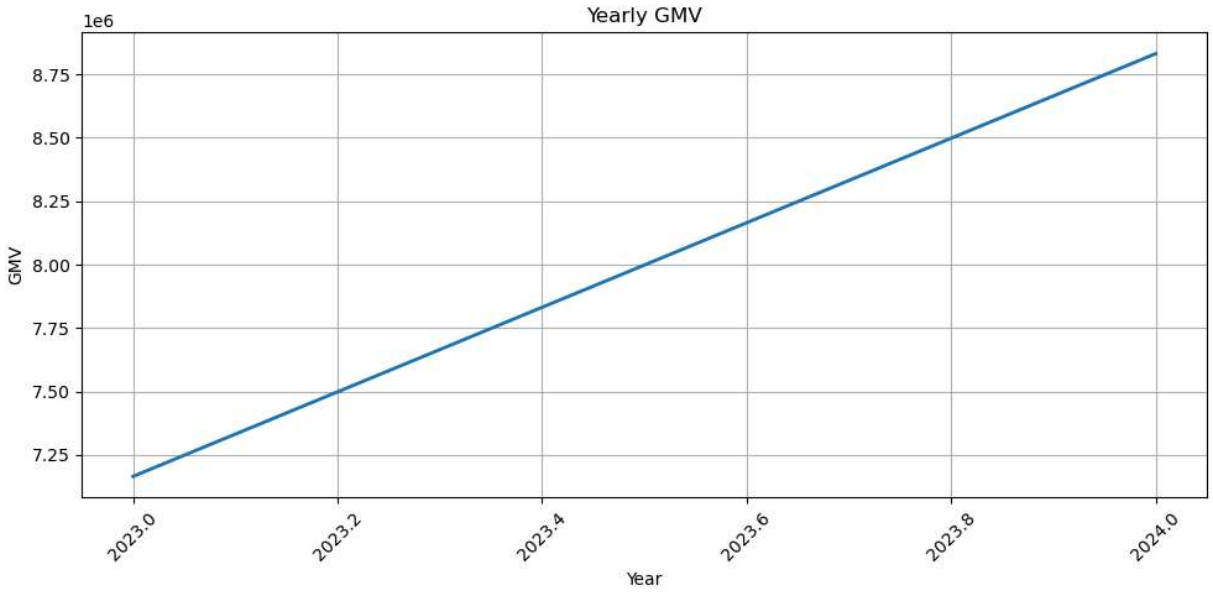
```
 * mysql+pymysql://root:***@localhost/db_netease
20 rows affected.
Returning data to local variable monthly_GMV
```

In [14]:
```
%%sql
daily_GMV <<
SELECT
d Date,
sum(payment_value) DGMV from dwd_netease_order_detail b1
LEFT JOIN o_time b2 on b1.customer_unique_id = b2.customer_unique_id
GROUP BY d
ORDER BY d;
```

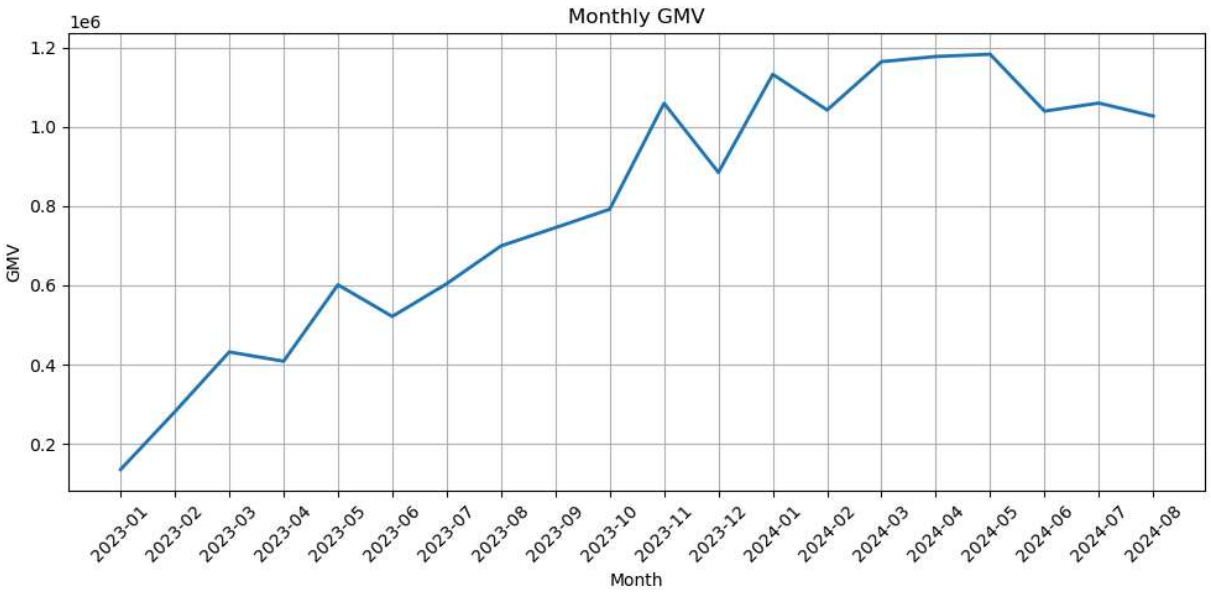
```
 * mysql+pymysql://root:***@localhost/db_netease
602 rows affected.
Returning data to local variable daily_GMV
```

In [15]:
```
def plot(data,Title,xlabel,ylabel,lwd=2):
    df = data.DataFrame()
    plt.figure(figsize=(10, 5))
    plt.plot(df.iloc[:,0], df.iloc[:,1], linestyle='-', linewidth=lwd)
    plt.title(Title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()
```

In [16]:
```
plot(yearly_GMV,'Yearly GMV','Year','GMV')
plt.show()
```



In [17]:
```
plot(monthly_GMV,'Monthly GMV','Month','GMV')
plt.show()
```



In [96]:
```
from openai import OpenAI
dgm = daily_GMV.DataFrame().to_json()

client = OpenAI()

completion = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a professional data and business ana
        {
            "role": "user",
            "content": f"Analysis the json data{dgm}. This data is the monthly gmv
        }
    ]
)
```

In [97]:
```
from IPython.display import Markdown,display
answer = completion.choices[0].message.content
answer = answer.replace("$"," \\$")
answer = answer.replace("**","")
display(Markdown(answer[11:]))
```

the monthly GMV (Gross Merchandise Volume) data for Temu represented in the provided JSON, we first need to extract and structure the key information for analysis. Below are the steps and insights derived from the analysis:

Data Overview

1. Structure: The data consists of two main attributes:
    - Date: This corresponds to timestamps representing the starting date of each month, starting from January 1, 2023.
    - DGMV: This represents the monthly GMV in dollars over a period of 601 months.
2. Frequency: The data is organized monthly, indicating how much revenue was generated from transactions on Temu each month over a plain timeframe.
3. Trend Analysis:
    - By examining the GMV trend visually, we can identify periods of significant growth, decline, and seasonality.
    - Additionally, calculating growth rates month-on-month can indicate how well the platform is performing over time.

Insights from GMV Analysis

1. Growth Patterns:
    - Determine the month-over-month growth percentages using:
        - [(GMV_{current month} - GMV_{previous month}) / GMV_{previous month} \times 100]
    - Identify months with the highest and lowest GMV and investigate potential reasons (e.g., marketing campaigns, sales events).
2. Seasonality:
    - Assess GMV data for peak seasons (e.g., holidays like Black Friday, Christmas, or Back to School).
    - Identify trends that repeat annually, if applicable, accounting for market behaviors.
3. Comparison with Industry Benchmarks:
    - Compare the GMV growth rates to competitors and industry standards to evaluate performance relative to the market.

Marketing Techniques to Leverage GMV Insights

To effectively utilize the insights derived from the GMV data, consider implementing the following marketing techniques:

1. Performance-Based Advertising:

- Utilize targeted advertising based on peak months with higher GMV to maximize ad spend effectiveness.

2. Seasonal Campaigns:

- Create promotional events tied to identified peak times (e.g., holidays, seasons). Use targeted email blasts and social media campaigns to alert customers of upcoming sales.

3. Customer Retargeting:

- Implement retargeting campaigns for previous customers, especially after significant purchases or during high GMV periods to encourage repeat sales.

4. Influencer Partnerships:

- Collaborate with influencers during months where GMV tends to peak to capitalize on their reach and drive traffic.

5. Data-Driven Content Marketing:

- Create content (blog posts, videos) around best-selling products or categories shown in the GMV data. Use SEO strategies to attract organic traffic.

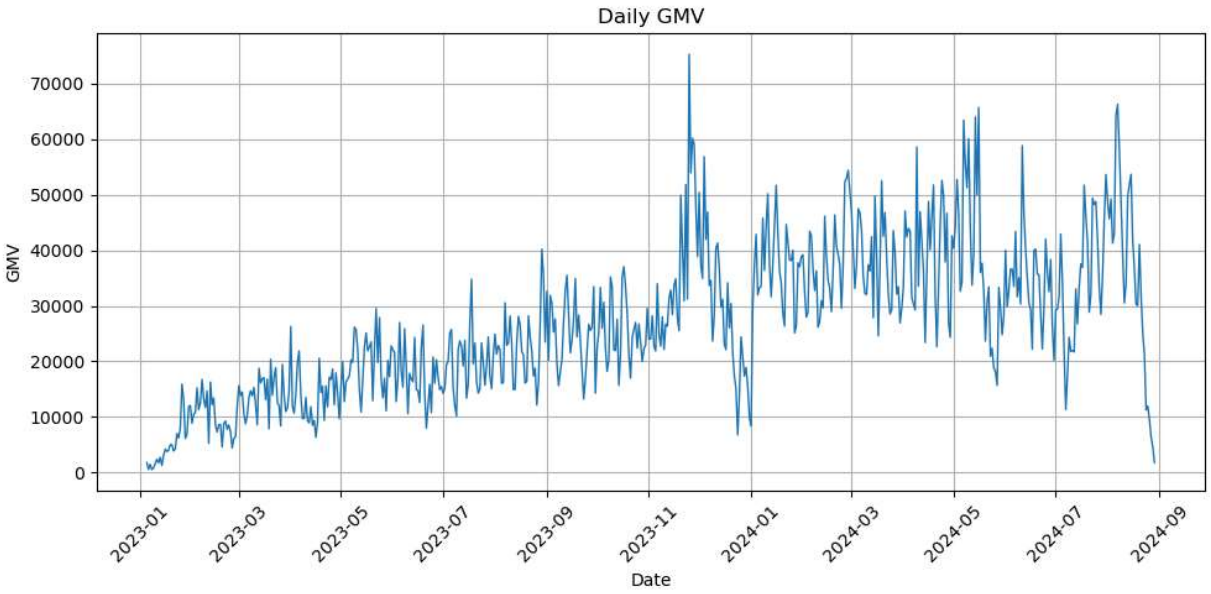
6. Sales Promotions:

- Develop loyalty programs or discounts based on customer purchase behavior reflected in the GMV data. Implement strategies to engage customers during slower months, flipping that trend back to growth.

Conclusion

By understanding the GMV performance trends of Temu at both a macro (monthly/yearly performance) and granular (individual product/category performance) level, you can develop targeted marketing strategies designed to foster growth, enhance customer retention, and optimize advertising spend. Regularly revisiting the data to adapt marketing techniques according to consumer behavior shifts is vital for sustained success.

In [153…
```
plot(daily_GMV,'Daily GMV','Date','GMV',1)
plt.show()
```



1.3 ARPU Analysis

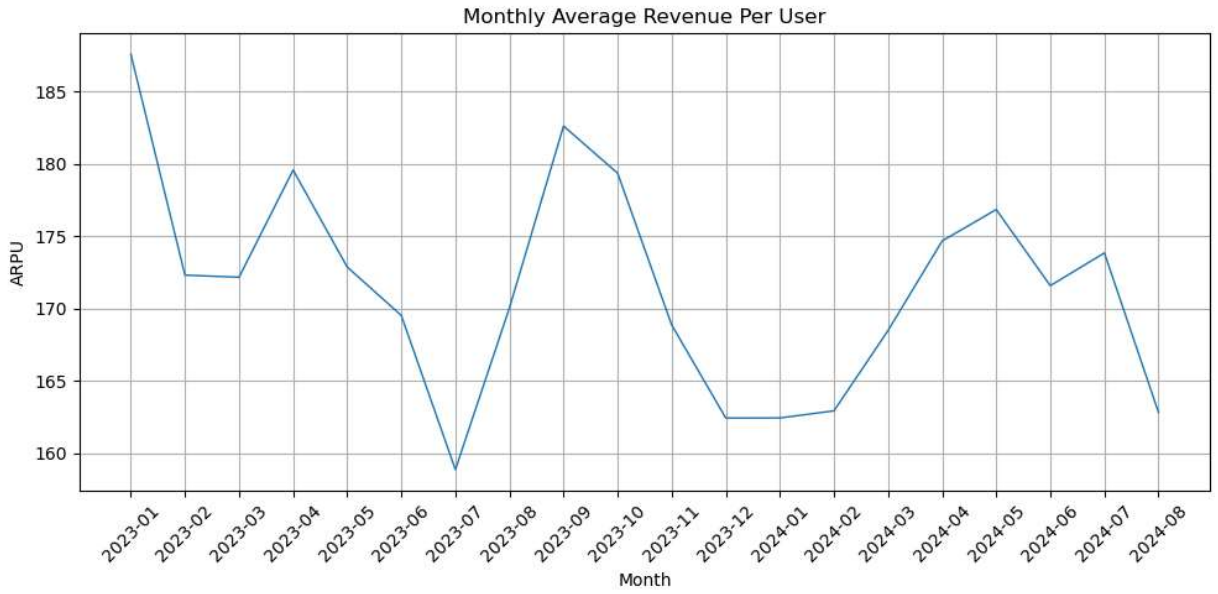
In [132…
```
%%sql

ya <<
SELECT  case
when m<10 then concat(y,'-0',m)
when m>=10 then concat(y,'-',m)
end as ym,
ROUND((sum(c1.payment_value)/count(DISTINCT c1.customer_unique_id)),2) m_ARPU
from dwd_netease_order_detail c1 LEFT JOIN o_time c2
on c1.customer_unique_id = c2.customer_unique_id
GROUP BY case
when m<10 then concat(y,'-0',m)
when m>=10 then concat(y,'-',m)
end
ORDER BY ym;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable ya
```

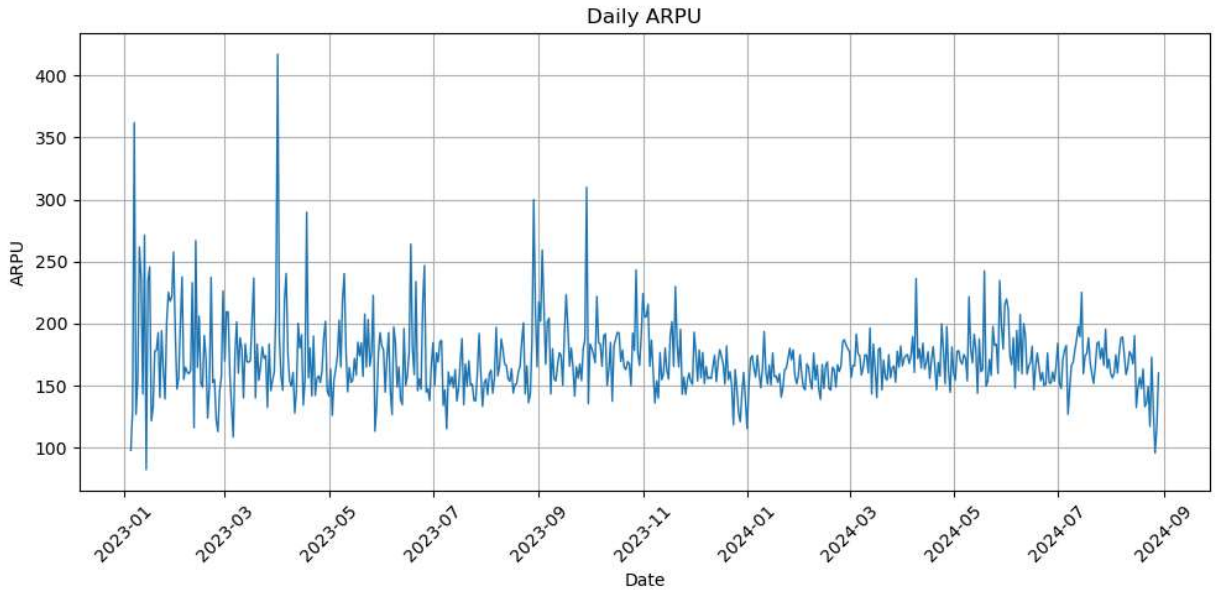
In [133…
```
%%sql

da <<
SELECT d Date,
ROUND((sum(c1.payment_value)/count(DISTINCT c1.customer_unique_id)),2) 日ARPU
from dwd_netease_order_detail c1 LEFT JOIN o_time c2
on c1.customer_unique_id = c2.customer_unique_id
GROUP BY d
ORDER BY d;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable da
```

In [155…
```
plot(ya,'Monthly Average Revenue Per User','Month','ARPU',1)
plt.show()
```



In [156…
```
plot(da,'Daily ARPU','Date','ARPU',1)
plt.show()
```



1.3 Active User Analysis

In [138…
```
%%sql

dau <<
SELECT d,count(distinct customer_unique_id) DAU
from o_time
GROUP BY d
ORDER BY d;
```

```
   mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable dau
```

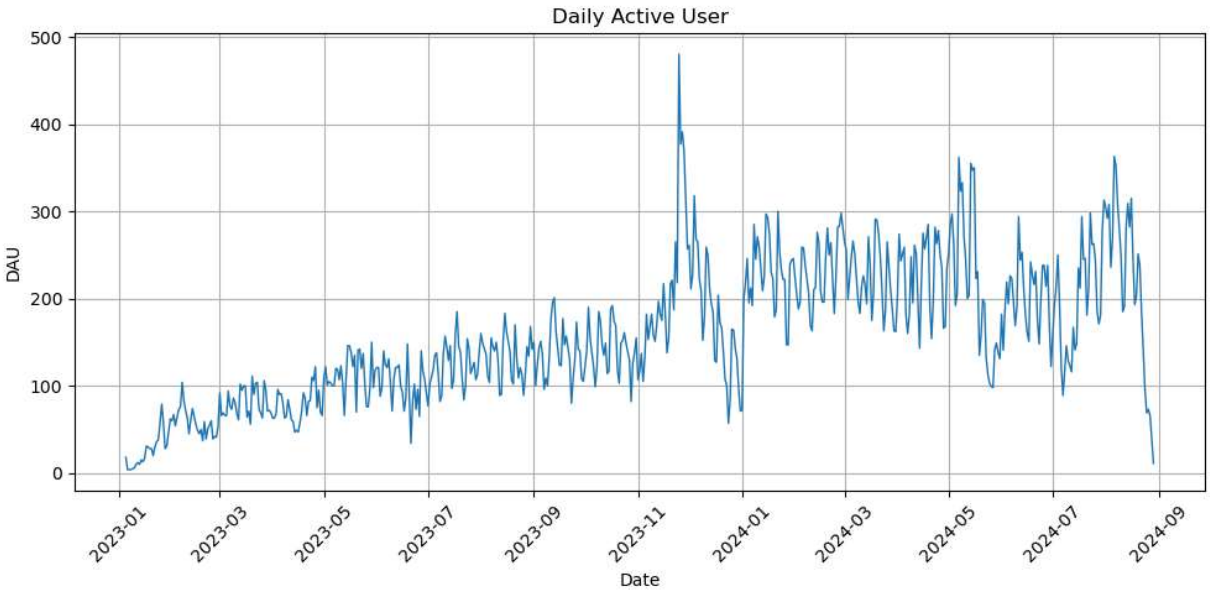
In [139…

```
%%sql

mau <<
SELECT
case
when m<10 then concat(y,'-0',m)
when m>=10 then concat(y,'-',m)
end as 年月,
count(DISTINCT customer_unique_id)MAU
from o_time
GROUP BY case
when m<10 then concat(y,'-0',m)
when m>=10 then concat(y,'-',m)
end;
```

```
   mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable mau
```

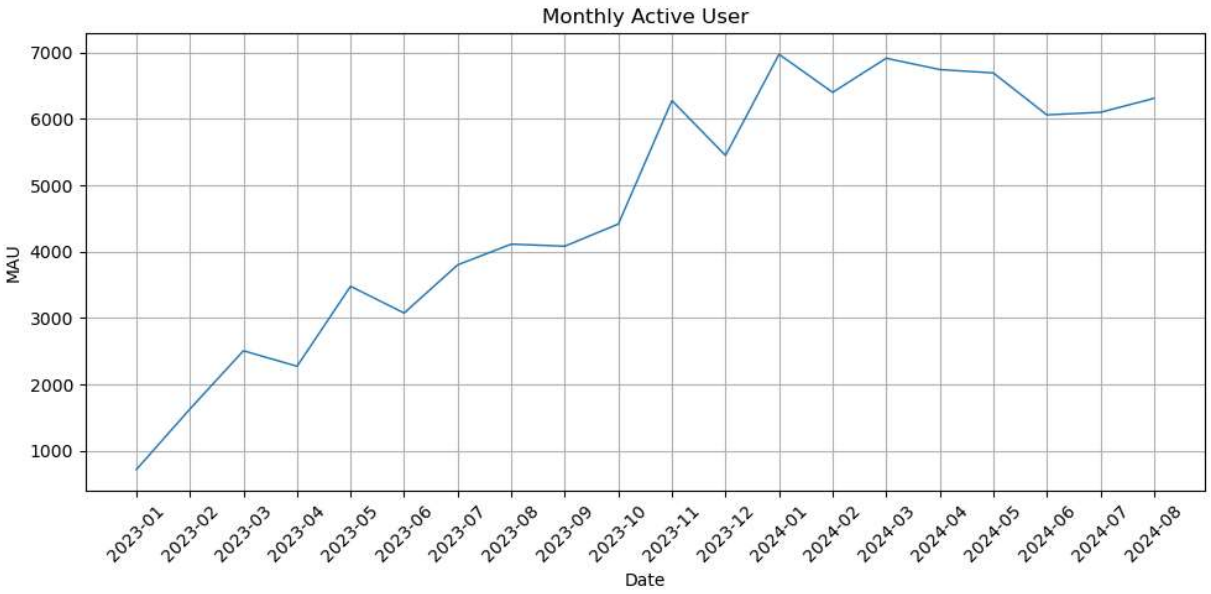
In [157…

```
plot(dau,'Daily Active User','Date','DAU',1)
plt.show()
```



In [158…

```
plot(mau,'Monthly Active User','Date','MAU',1)
plt.show()
```

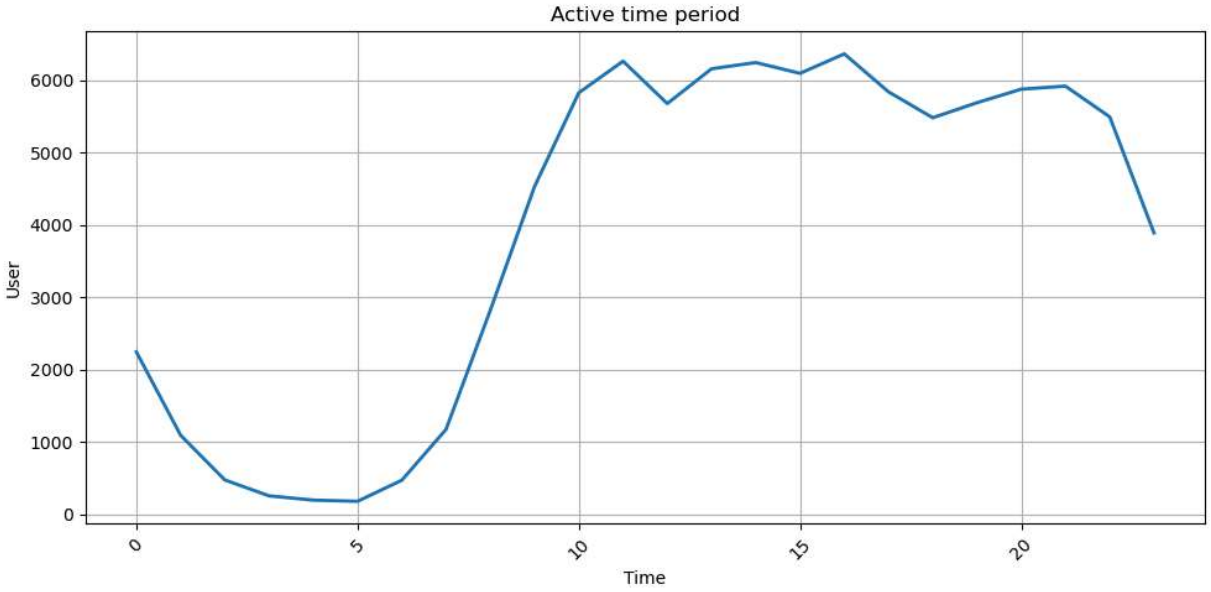


Active time period

In [160…
```sql
%%sql
Atp <<
SELECT h 时段,
count(DISTINCT customer_unique_id) 时段活跃用户数
from o_time
GROUP BY h
ORDER BY h;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable Atp
```

In [161…
```
plot(Atp,'Active time period', 'Time','User')
plt.show()
```



2. User Analysis

2.1 RFM modeling

In [213…

```sql
%%sql
-- R
-- If most recent purchase frequency is more than his or her avg fretency

drop view if exists Recency;
CREATE VIEW Recency AS
SELECT
  customer_unique_id,
  CASE
    WHEN DATEDIFF(CURDATE(),MAX(order_purchase_timestamp)) > AVG(DATEDIFF(order_pur
    THEN 1
    ELSE 0
  END AS R
FROM (
  SELECT
    customer_unique_id,
    order_purchase_timestamp,
    LAG(order_purchase_timestamp) OVER (PARTITION BY customer_unique_id ORDER BY or
  FROM dwd_netease_order_detail
) a
GROUP BY customer_unique_id;



-- F
-- If frequency is more than his or her avg frequency
DROP VIEW IF EXISTS Frequency;
CREATE VIEW Frequency AS
SELECT
    customer_unique_id,
    CASE
        WHEN COUNT(1) > (SELECT AVG(order_cnt)
                         FROM (SELECT customer_unique_id, COUNT(*) AS order_cnt
                               FROM dwd_netease_order_detail
                               GROUP BY customer_unique_id) AS a)
        THEN 1
        ELSE 0
    END AS F
FROM dwd_netease_order_detail
GROUP BY customer_unique_id;



-- M
-- If the money spent is more than his or her avg spending
DROP VIEW IF EXISTS Monetary;
CREATE VIEW Monetary AS
SELECT
```

```
    customer_unique_id,
    CASE
        WHEN SUM(payment_value) > (SELECT AVG(payment_value_summ)
                                   FROM (SELECT customer_unique_id, SUM(payment_val
                                         FROM dwd_netease_order_detail
                                         GROUP BY customer_unique_id) AS a)
        THEN 1
        ELSE 0
    END AS M
FROM dwd_netease_order_detail
GROUP BY customer_unique_id;


-- Final RFM model
drop view if exists rfm;
CREATE VIEW rfm AS
SELECT
Recency.customer_unique_id,
(CASE
WHEN R=0 AND F=1 AND M=1 THEN "High-value users"
WHEN R=0 AND F=0 AND M=1 THEN "High-potential users"
WHEN R=1 AND F=1 AND M=1 THEN "High-maintain users"
WHEN R=1 AND F=0 AND M=1 THEN "High-retention users"
WHEN R=0 AND F=1 AND M=0 THEN "Regular-value users"
WHEN R=0 AND F=0 AND M=0 THEN "Regular-potential users"
WHEN R=1 AND F=1 AND M=0 THEN "Regular-maintain users"
WHEN R=1 AND F=0 AND M=0 THEN "Regular-retention users"
ELSE "Others" END) AS User_Grade
FROM Recency
INNER JOIN Frequency
ON Recency.customer_unique_id = Frequency.customer_unique_id
INNER JOIN Monetary
ON Recency.customer_unique_id = Monetary.customer_unique_id;
```

```
  mysql+pymysql://root:***@localhost
* mysql+pymysql://root:***@localhost/db_netease
```

Out[213… []

2.2 Distribution of User types

In [214…

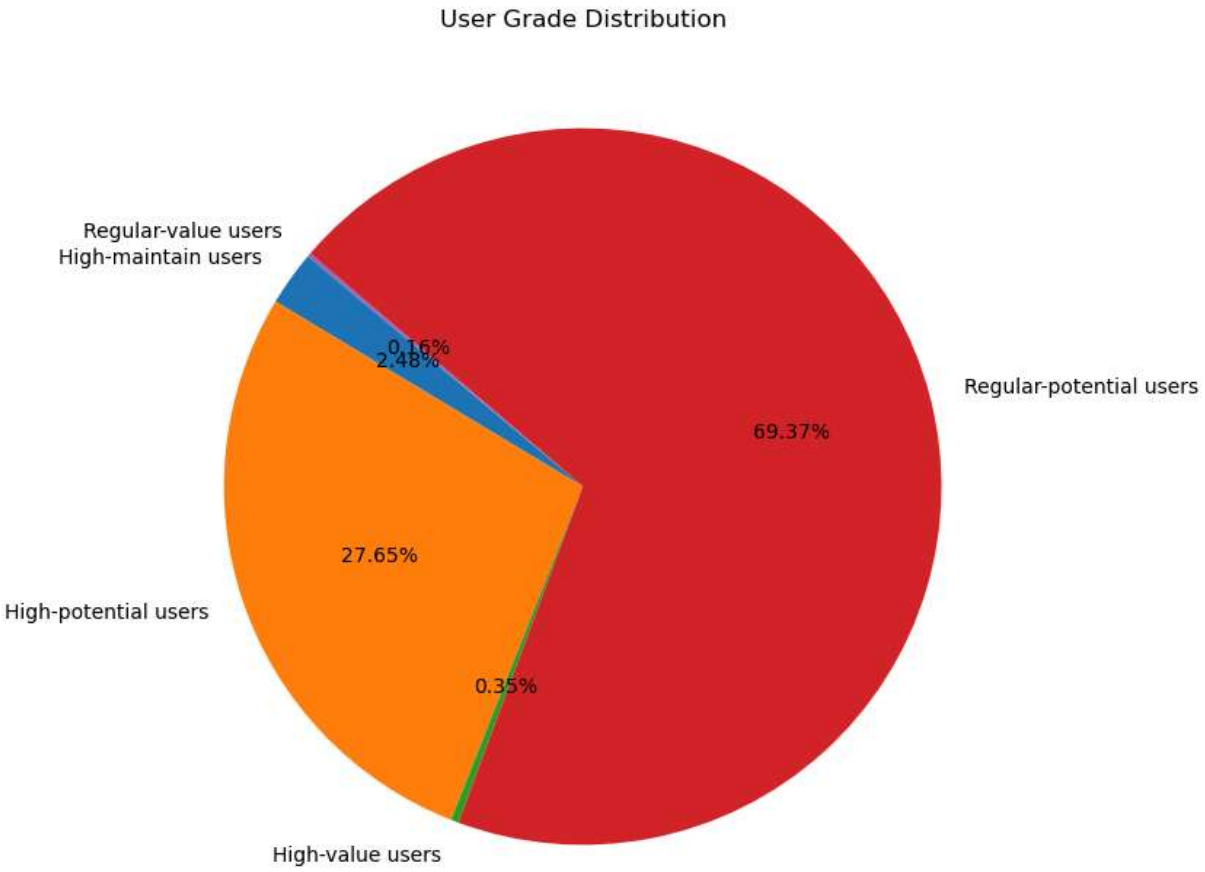
```
%%sql
gc <<
SELECT User_Grade,count(*) User_count,
count(*)/(SELECT count(*) from rfm) Percentage
from rfm
GROUP BY User_Grade
ORDER BY User_Grade;
```

```
  mysql+pymysql://root:***@localhost
* mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable gc
```

In [187…

```
df = gc.DataFrame()
plt.figure(figsize=(8, 8))
plt.pie(
```

```
    df['User_count'],
    labels=df['User_Grade'],
    autopct='%1.2f%%',
    startangle=140
)
plt.title('User Grade Distribution')
plt.show()
```



In [188…

```
df['Percentage'] = df['Percentage'].map(lambda x: round(x*100, 2))
df
```

Out[188…

	User_Grade	User_count	Percentage
0	High-maintain users	2285	2.48
1	High-potential users	25498	27.65
2	High-value users	319	0.35
3	Regular-potential users	63975	69.37
4	Regular-value users	144	0.16

There is no high retention users, regular retention and maintain users in the output.

2.2 Product Preference of Users

In [197…
```
%%sql

pou1 <<
SELECT
User_Grade,product_category_name Category,
sum_payment_value Spending,
rank() over(order by sum_payment_value desc) ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'High-value users'
        GROUP BY User_Grade,product_category_name
)a

;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou1
```

In [198…
```
%%sql

pou2 <<
SELECT
User_Grade,product_category_name Category,
sum_payment_value Spending,
rank() over(order by sum_payment_value desc)ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'High-potential users'
        GROUP BY User_Grade,product_category_name
)a
;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou2
```

In [200…
```
%%sql

pou3 <<
SELECT
User_Grade,product_category_name Category,
sum_payment_value Spending,
```

```
rank() over(order by sum_payment_value desc)ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'High-maintain users'
        GROUP BY User_Grade,product_category_name
)a
;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou3
```

In [207…

```
%%sql

pou4 <<
SELECT
User_Grade,product_category_name Category,
sum_payment_value Spending,
rank() over(order by sum_payment_value desc)ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'High-retention users'
        GROUP BY User_Grade,product_category_name
)a
;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou4
```

In [202…

```
%%sql

pou5 <<
SELECT
User_Grade,product_category_name Category,
sum_payment_value Spending,
rank() over(order by sum_payment_value desc)ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'Regular-value users'
        GROUP BY User_Grade,product_category_name
```

```
)a
;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou5
```

In [203…

```
%%sql

pou6 <<
SELECT
User_Grade,product_category_name Category,
sum_payment_value Spending,
rank() over(order by sum_payment_value desc)ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'Regular-potential users'
        GROUP BY User_Grade,product_category_name
)a
;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou6
```

In [204…

```
%%sql

pou7 <<
SELECT
User_Grade,product_category_name Category,
sum_payment_value Spending,
rank() over(order by sum_payment_value desc)ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'Regular-maintain users'
        GROUP BY User_Grade,product_category_name
)a
;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou7
```

In [206…

```
%%sql

pou8 <<
SELECT
```

```
User_Grade,product_category_name Category,
sum_payment_value Spending,
rank() over(order by sum_payment_value desc)ranks
FROM
(
        select
        User_Grade,product_category_name,
        sum(a.payment_value) sum_payment_value
        FROM dwd_netease_order_detail a
        JOIN rfm b on a.customer_unique_id = b.customer_unique_id
        where User_Grade = 'Regular-retention users'
        GROUP BY User_Grade,product_category_name
)a
;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pou8
```

In [218…

```
import os
os.makedirs('Output_Excels', exist_ok=True)

writer = pd.ExcelWriter('User Preference.xlsx')
for p in [pou1,pou2,pou3,pou4,pou5,pou6,pou7,pou8]:
    df = p.DataFrame()
    if df.empty:
        continue
    print(df.head(5))
    df.to_excel(writer, sheet_name = f'{df.iloc[0,0]}.xlsx')
writer.close()
```

```
         User_Grade                Category  Spending  ranks
0  High-value users      relogios_presentes  12171.72      1
1  High-value users         cama_mesa_banho  11158.20      2
2  High-value users            beleza_saude  10493.27      3
3  High-value users  informatica_acessorios   9532.68      4
4  High-value users           esporte_lazer   8547.68      5
            User_Grade               Category   Spending  ranks
0  High-potential users      relogios_presentes  913453.29      1
1  High-potential users            beleza_saude  841989.91      2
2  High-potential users  informatica_acessorios  615555.21      3
3  High-potential users           esporte_lazer  577367.90      4
4  High-potential users         cama_mesa_banho  526541.64      5
           User_Grade               Category  Spending  ranks
0  High-maintain users         cama_mesa_banho  96225.96      1
1  High-maintain users           esporte_lazer  69690.75      2
2  High-maintain users        moveis_decoracao  66232.94      3
3  High-maintain users  informatica_acessorios  56851.32      4
4  High-maintain users            beleza_saude  49730.29      5
           User_Grade              Category  Spending  ranks
0  Regular-value users        cama_mesa_banho   2645.33      1
1  Regular-value users          esporte_lazer   1476.17      2
2  Regular-value users       moveis_decoracao   1474.17      3
3  Regular-value users           beleza_saude   1125.62      4
4  Regular-value users  utilidades_domesticas   1102.68      5
               User_Grade               Category   Spending  ranks
0  Regular-potential users         cama_mesa_banho  548709.85      1
1  Regular-potential users            beleza_saude  475725.65      2
2  Regular-potential users           esporte_lazer  436052.60      3
3  Regular-potential users        moveis_decoracao  344669.65      4
4  Regular-potential users  informatica_acessorios  328775.94      5
```

2.3 Demographic Information

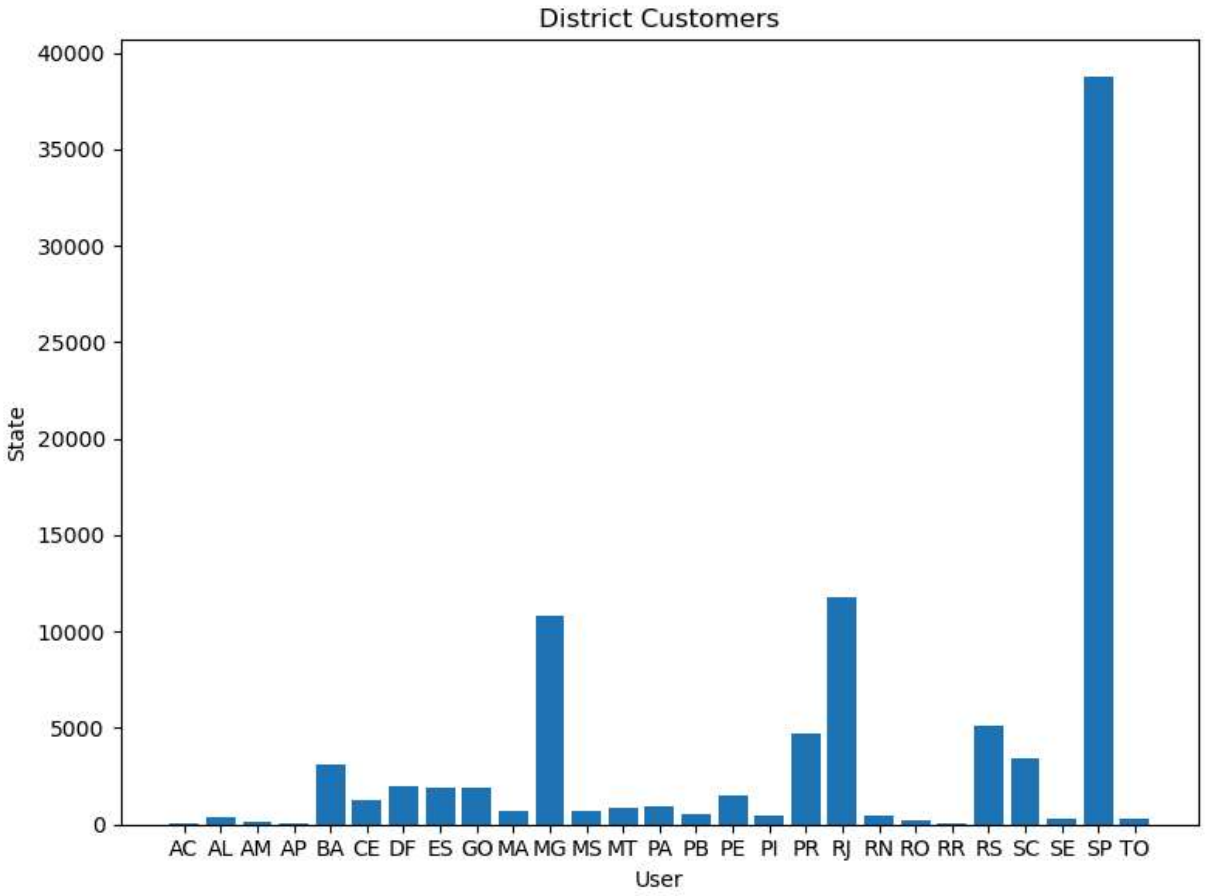
In [220…

```
%%sql

disc  <<

SELECT customer_state as District,count(DISTINCT customer_unique_id)
as Customer
from dwd_netease_order_detail
GROUP BY customer_state;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable disc
```

In [225…

```
df = disc.DataFrame()
plt.figure(figsize=(8, 6))
plt.bar(df['District'], df['Customer'])
plt.xlabel('User')
plt.ylabel('State')
plt.title('District Customers')
plt.tight_layout()
plt.show()
```



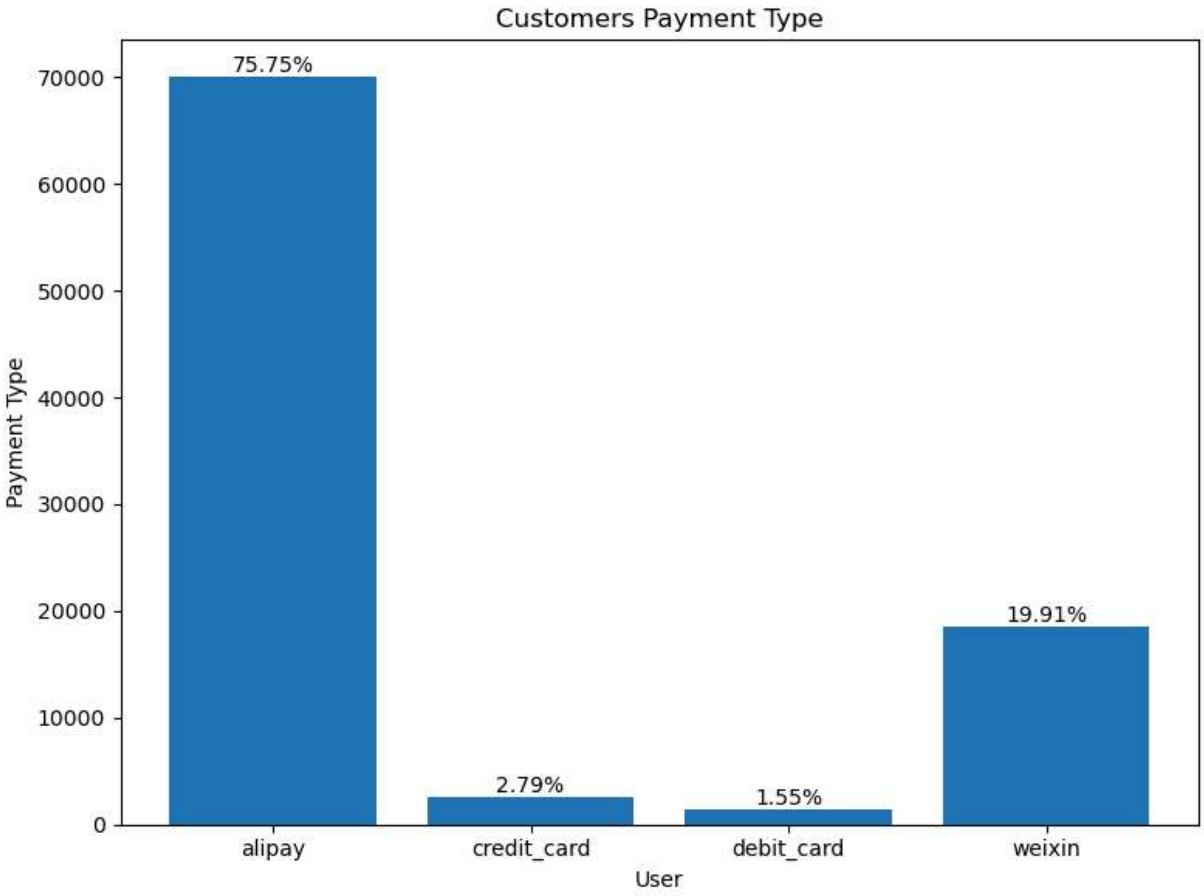
2.4 Payment Type of Users

In [227…
```
%%sql

pt <<

SELECT payment_type as Pay_t,count(DISTINCT customer_unique_id) as Users,
count(1)/(select count(1) from dwd_netease_order_detail) as Percentage
from dwd_netease_order_detail
GROUP BY payment_type;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pt
```

In [232…
```
df = pt.DataFrame()
plt.figure(figsize=(8, 6))
bars = plt.bar(df['Pay_t'], df['Users'], color='blue')

# Add percentage labels on top of each bar
for bar, percentage in zip(bars, df['Percentage']):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f'{round(percentage*100,2)}%',
        ha='center',
        va='bottom'
```

```
    )
plt.bar(df['Pay_t'], df['Users'])
plt.xlabel('User')
plt.ylabel('Payment Type')
plt.title('Customers Payment Type')
plt.tight_layout()
plt.show()
```



2.5 Installment Period

In [239…

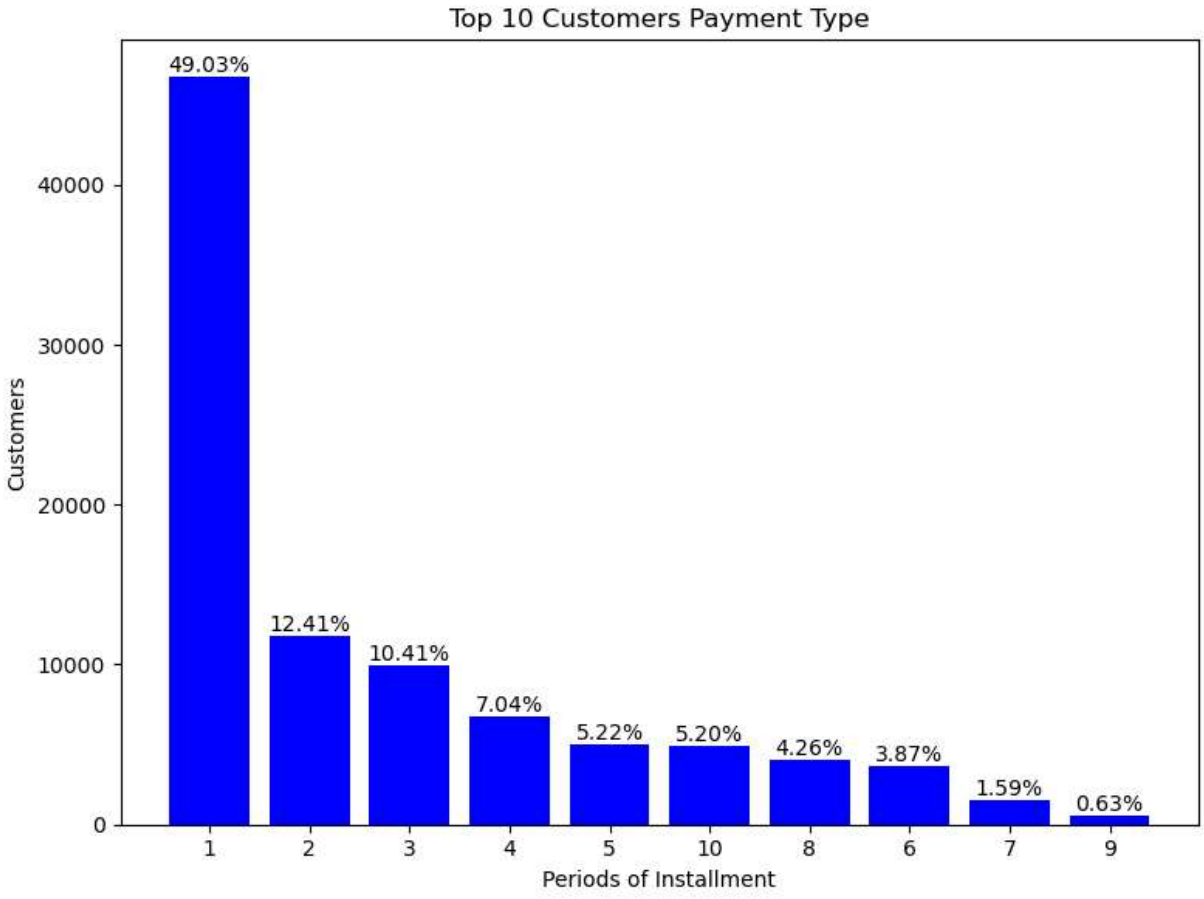
```
%%sql

ip <<
SELECT
payment_installments as Period,count(1) as Customer,
round(count(1)*100/(SELECT count(1) from dwd_netease_order_detail),2) as Percentage
from dwd_netease_order_detail
GROUP BY payment_installments
order by Customer desc;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable ip
```

In [246…

```
df = ip.DataFrame().head(10)
plt.figure(figsize=(8, 6))
bars = plt.bar(df['Period'], df['Customer'], color='blue')
```

```
# Add percentage labels on top of each bar
for bar, percentage in zip(bars, df['Percentage']):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f'{percentage}%',
        ha='center',
        va='bottom'
    )
plt.xlabel('Periods of Installment')
plt.ylabel('Customers')
plt.title('Top 10 Customers Payment Type')
plt.tight_layout()
plt.show()
```



3. Seller Analysis

3.1 Seller district analysis

In [248…

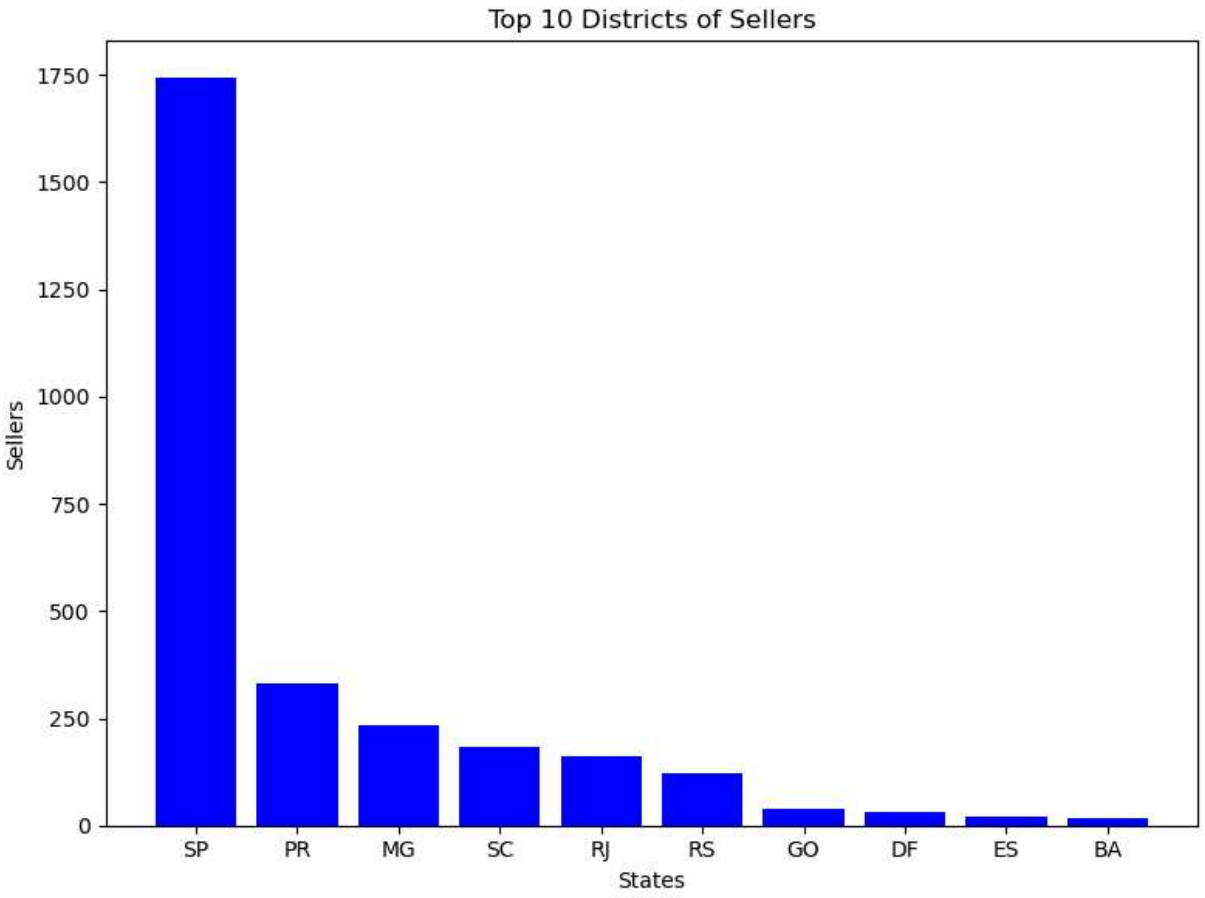
```
%%sql

sdc <<
select
 seller_state as District,
 count(distinct seller_id) as Seller
```

```
from dwd_netease_order_detail
GROUP BY seller_state
order by Seller desc;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable sdc
```

In [250…

```
df = sdc.DataFrame().head(10)
plt.figure(figsize=(8, 6))
bars = plt.bar(df['District'], df['Seller'], color='blue')
plt.xlabel('States')
plt.ylabel('Sellers')
plt.title('Top 10 Districts of Sellers')
plt.tight_layout()
plt.show()
```



3.2 Merchant transaction interval

In [254…

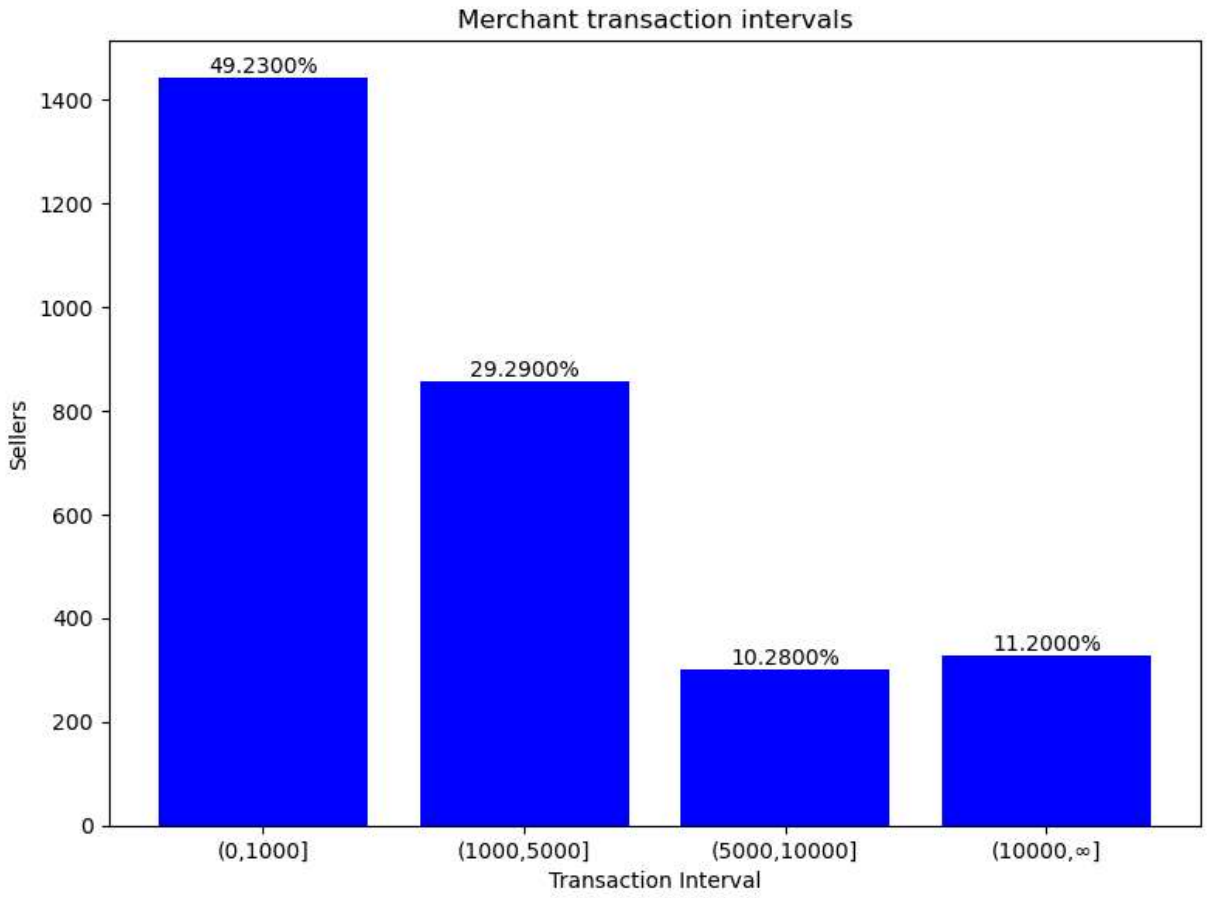
```
%%sql
ti <<
SELECT Intv,count(1) as Seller,
count(1)/(SELECT count(DISTINCT seller_id)from dwd_netease_order_detail)as Percenta
from (SELECT seller_id,sum(payment_value) as sum_pa,
case when sum(payment_value) < 1000 then '(0,1000]'
                when sum(payment_value) > 1000 and sum(payment_value)<=5000
                then '(1000,5000]'
                when sum(payment_value)>5000 and sum(payment_value)<=10000
```

```
                then '(5000,10000]'
                else '(10000,∞]' end as 'Intv' from dwd_netease_order_detail
GROUP BY seller_id
order by sum_pa) a
GROUP BY Intv;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable ti
```

In [258…

```
df = ti.DataFrame()
plt.figure(figsize=(8, 6))
bars = plt.bar(df['Intv'], df['Seller'], color='blue')

# Add percentage labels on top of each bar
for bar, percentage in zip(bars, df['Percentage']):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f'{percentage*100}%',
        ha='center',
        va='bottom'
    )
plt.xlabel('Transaction Interval')
plt.ylabel('Sellers')
plt.title('Merchant transaction intervals')
plt.tight_layout()
plt.show()
```



3.3 Seller Rating Analysis

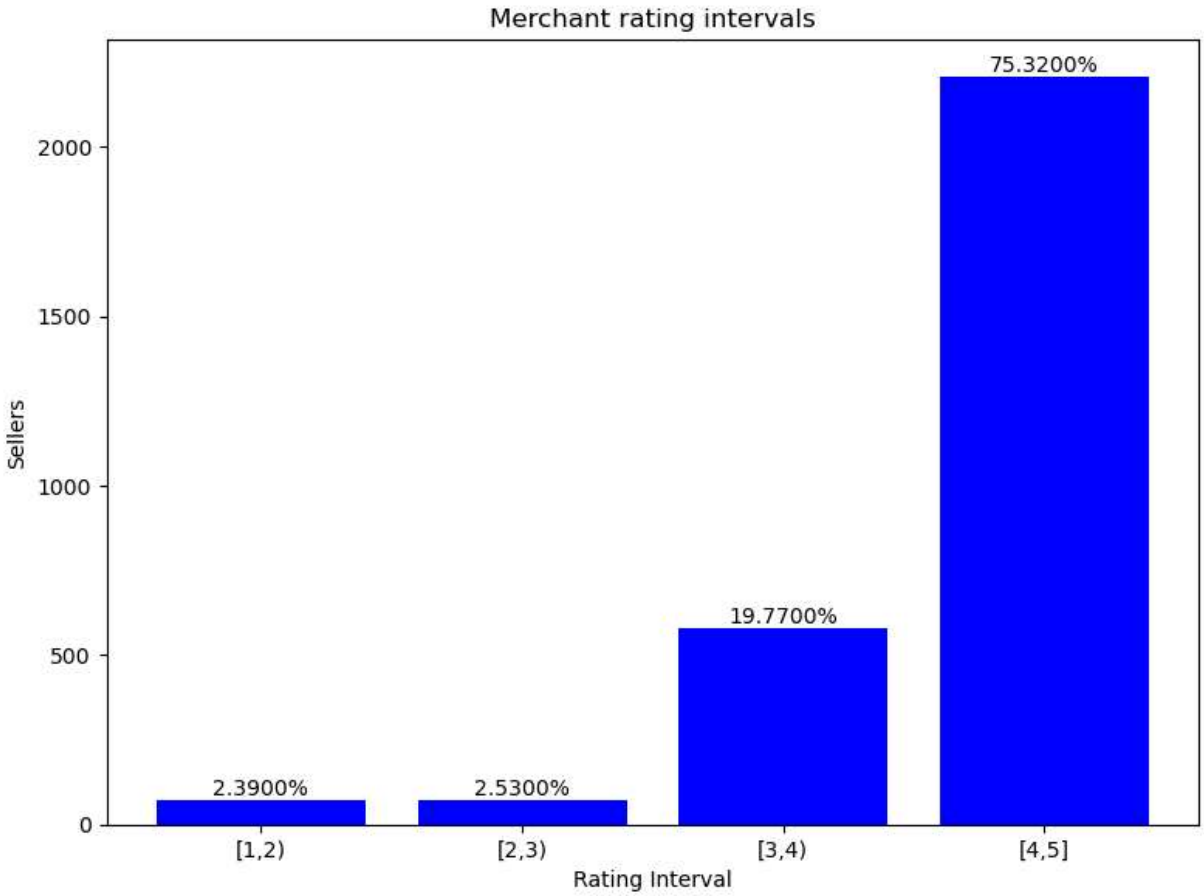
In [273…

```
%%sql

ra <<
SELECT rating,count(1) as Seller,
count(1)/(SELECT count(DISTINCT seller_id)from dwd_netease_order_detail)as percenta
from(SELECT seller_id,avg(review_score) as avg_,
case when avg(review_score)>= 1 and avg(review_score)< 2 then '[1,2)'
     when avg(review_score)>= 2 and avg(review_score)< 3 then '[2,3)'
                when avg(review_score)>= 3 and avg(review_score)< 4 then '[3,4)'
                else '[4,5]' end as 'rating'
 from dwd_netease_order_detail
 GROUP BY seller_id
 order by avg_
)a
group by rating;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable ra
```

In [274…

```
df = ra.DataFrame()
plt.figure(figsize=(8, 6))
bars = plt.bar(df['rating'], df['Seller'], color='blue')

# Add percentage labels on top of each bar
for bar, percentage in zip(bars, df['percentage']):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f'{percentage*100}%',
        ha='center',
        va='bottom'
    )
plt.xlabel('Rating Interval')
plt.ylabel('Sellers')
plt.title('Merchant rating intervals')
plt.tight_layout()
plt.show()
```



4. Product Analysis

Product Sales Revenue, Volume, and Unit Price

In [276…
```
%%sql
pi <<
SELECT
        case when product_category_name !='' then product_category_name
        else 'lost_name' end Category,
        sum(payment_value) Revenue,
        sum(order_item_id) Volume,
        sum(payment_value)/sum(order_item_id) Unit_Price
from dwd_netease_order_detail
GROUP BY product_category_name;
```

```
  mysql+pymysql://root:***@localhost
 * mysql+pymysql://root:***@localhost/db_netease
Returning data to local variable pi
```

In [306…
```
df = pi.DataFrame().round(2)
df.sort_values(by=['Revenue'], ascending=False)
```

Out[306…

	Category	Revenue	Volume	Unit_Price
8	beleza_saude	1379064.74	8902.0	154.92
4	relogios_presentes	1224308.75	5591.0	218.98
2	cama_mesa_banho	1185280.98	9956.0	119.05
13	esporte_lazer	1093135.10	7885.0	138.63
10	informatica_acessorios	1011717.90	7019.0	144.14
...	...	...	...	...
31	pc_gamer	1398.52	9.0	155.39
69	casa_conforto_2	1196.32	24.0	49.85
71	cds_dvds_musicais	954.99	13.0	73.46
63	fashion_roupa_infanto_juvenil	608.54	6.0	101.42
73	seguros_e_servicos	324.51	2.0	162.26

74 rows × 4 columns

4.2 Pareto Analysis of Product Categories: Identifying Categories Contributing Most to the Revenue

In [307…

```
total_revenue = df['Revenue'].sum()
total_volume = df['Volume'].sum()
df['Revenue_Percentage'] = (df['Revenue'] / total_revenue) * 100
df['Volume_Percentage'] = (df['Volume'] / total_volume) * 100
df = df[['Category', 'Revenue_Percentage', 'Volume_Percentage']]
df = df.sort_values(by=['Revenue_Percentage'], ascending=False)
df['Cumulative_Revenue_Percentage'] = df['Revenue_Percentage'].cumsum()
df.round(2)
```

Out[307…

	Category	Revenue_Percentage	Volume_Percentage	Cumulative_Reven
8	beleza_saude	9.21	8.73	
4	relogios_presentes	8.17	5.48	
2	cama_mesa_banho	7.91	9.76	
13	esporte_lazer	7.30	7.73	
10	informatica_acessorios	6.75	6.88	
...	...	...	...	
31	pc_gamer	0.01	0.01	
69	casa_conforto_2	0.01	0.02	
71	cds_dvds_musicais	0.01	0.01	
63	fashion_roupa_infanto_juvenil	0.00	0.01	
73	seguros_e_servicos	0.00	0.00	

74 rows × 4 columns

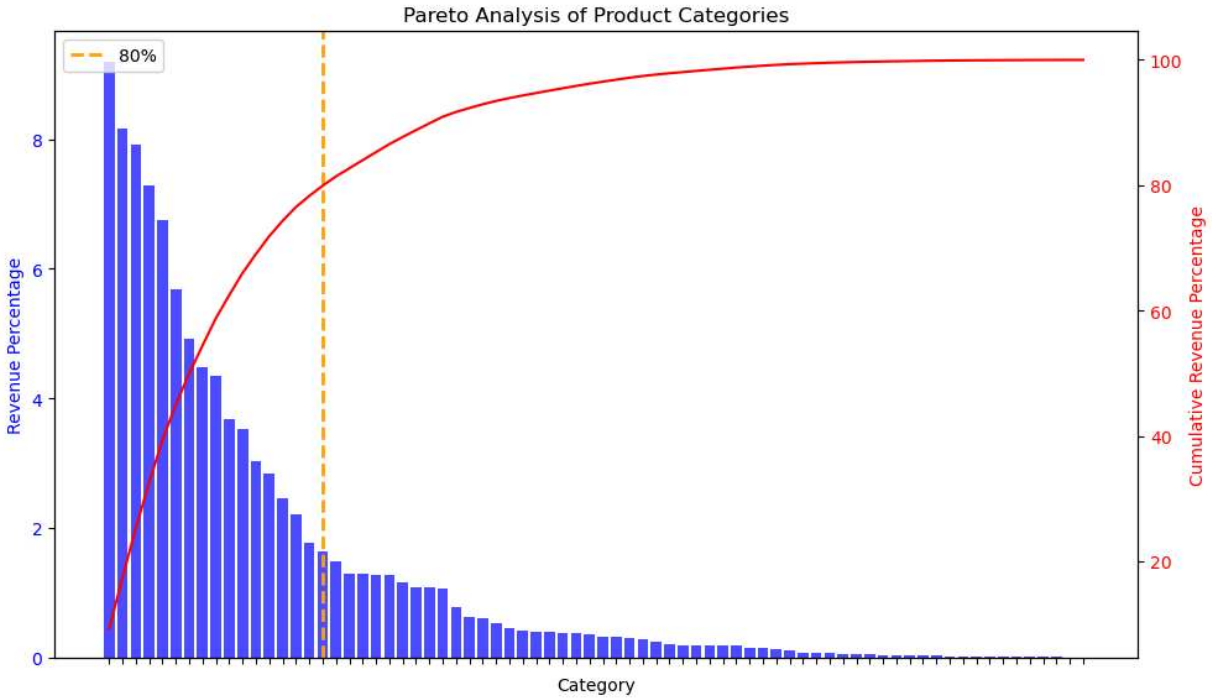
In [321…

```
cnt = len(df[df['Cumulative_Revenue_Percentage']< 80])
print(f'Top {cnt}th companies gained 80% of revenue.')
```

```
Top 17th companies gained 80% of revenue.
```

In [325…

```
fig, ax1 = plt.subplots(figsize=(10, 6))

# Bar plot for Revenue Percentage
ax1.bar(df['Category'], df['Revenue_Percentage'], color='blue', alpha=0.7)
ax1.set_xlabel('Category')
ax1.set_ylabel('Revenue Percentage', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Line plot for Cumulative Revenue Percentage
ax2 = ax1.twinx()
ax2.plot(df['Category'], df['Cumulative_Revenue_Percentage'], color='red', linestyl
ax2.set_ylabel('Cumulative Revenue Percentage', color='red')
ax2.tick_params(axis='y', labelcolor='red')

ax1.axvline(x=16, color='orange', linestyle='--', linewidth=2, label='80%')
ax1.legend(loc='upper left')
# Add a title and improve layout
plt.title('Pareto Analysis of Product Categories')
plt.tight_layout()
ax1.set_xticklabels([])
# Show the plot
plt.show()
```



In [1]:
```
! jupyter nbconvert *.ipynb --to html
```

```
[NbConvertApp] Converting notebook sql_customer.ipynb to html
[NbConvertApp] WARNING | Alternative text is missing on 16 image(s).
[NbConvertApp] Writing 1358577 bytes to sql_customer.html
```

In [ ]:
```
```