Customer Segmentation Analysis

---

1. Project Introduction

In recent years, a lot of consumption data were accumulated from real order receipts to online order records, more and more offline-to-online shopping platforms started customizing group and targeting the customer with customer segmentation. E-commerce customer profiles and order data were utilized customize marketing and advertising strategy for merchants. In fact, with the assistance of customer segmentation analysis and recommendations, merchants can not only make their business goal clearly by understanding the structures and the audience of marketing, but also reduce marketing cost and improve conversion rate.

Customer Segmentation Charts using Power BI: **click>** (https://app.powerbi.com/view?r=eyJrIjoiNmRiMGVlMjMtODcwZi00NjZjLTg1NTgtY2E2YjQ1YjAyYTBmIiwidCI6ImU5N2Q5OTExLTY1OTEtNGNjM

2. Data collection

This report would carry out a detailed analysis on customer's online ordering data of the period from 2015-06-26 06:00:00 to 2016-10-31 23:00:00 from Alibaba transaction data: **Alibaba raw data** (https://tianchi.aliyun.com/dataset/dataDetail?dataId=58&userId=1&lang=en-us)

- 6967,4100 rows of historical transaction data of customer's orders
- 1958,3949 customers who had order behavior
- 2000 shops information including location and product categorys

3. Abstract

This report analyzes customer segmentation by looking into not only the customer's similar demographic, geographic trait, and behavior patterns, but also similar shopping interest, lifestyles by having the aid of online ordering data, as a result it can provide value-adding and cost-saving recommendation for different types of:

- Customer marketing targets a specific customer group with RFM segmentation cluster.
- Customer marketing targets a specific product category against the specific customers preferences group.
- Customer marketing targets a specific product group against the specific time and customer group.
- Customer marketing targets the top consumption level's city against the customer's orders.

The results were given after analyzing the dataset from different features, including:

- Customer's profile like consumption level, location.
- Customer's order habits, like order frequency, order recency. All other feature will support as many as possible audience.

The process include data gathering, cleaning, transforming, modeling, analysing and visualising. Although it is a simple project, it is covering all pharses of data analysis.

4.Report Assumption

- The report assumes that the transaction data for this report can represente the typical behaviors of the entire customers on Alibaba sufficiently.
- The report can be used as an effective insight to help merchants to identify customers segmentation and place marketing.
- The report takes the target customers as the key factor for making a marketing and advertising strategy. The other influencing factors that affect the effectiveness of advertising, such as delivery channels, conversion paths, and ads cost are not included at this stage.
- Marketing and ads teams are target audience for this report.

5. Problem Statement

The heart of e-commerce is finding the best suitable customers, products and marketplace, supporting the stakeholder on categorizing the customers and marketing focus. For the channel and marketing teams, it would be a tough identification of the most likely buyers of a company's product or service, and how much premium worth to be put into the target customers groups. Hence, the report presents findings by properly reformulating the problem.

6.Analysis Tools

- SQL:data cleaning, query, transformation and analysis.
- Power BI: data visualisation, ad-hoc reporting, and simple transformation.
- Python Pandas: data ingestion and simple transformation.
- Python: data loading and sampling.
- Docker: analysis environment deployment.
- Jupyter: data analysis and reporting.

Please refer to the deployment of PostgreSQL, Jupyter notebook on Docker
https://github.com/PengGitt/Customer_Segmentation/tree/main/docker
(https://github.com/PengGitt/Customer_Segmentation/tree/main/docker)

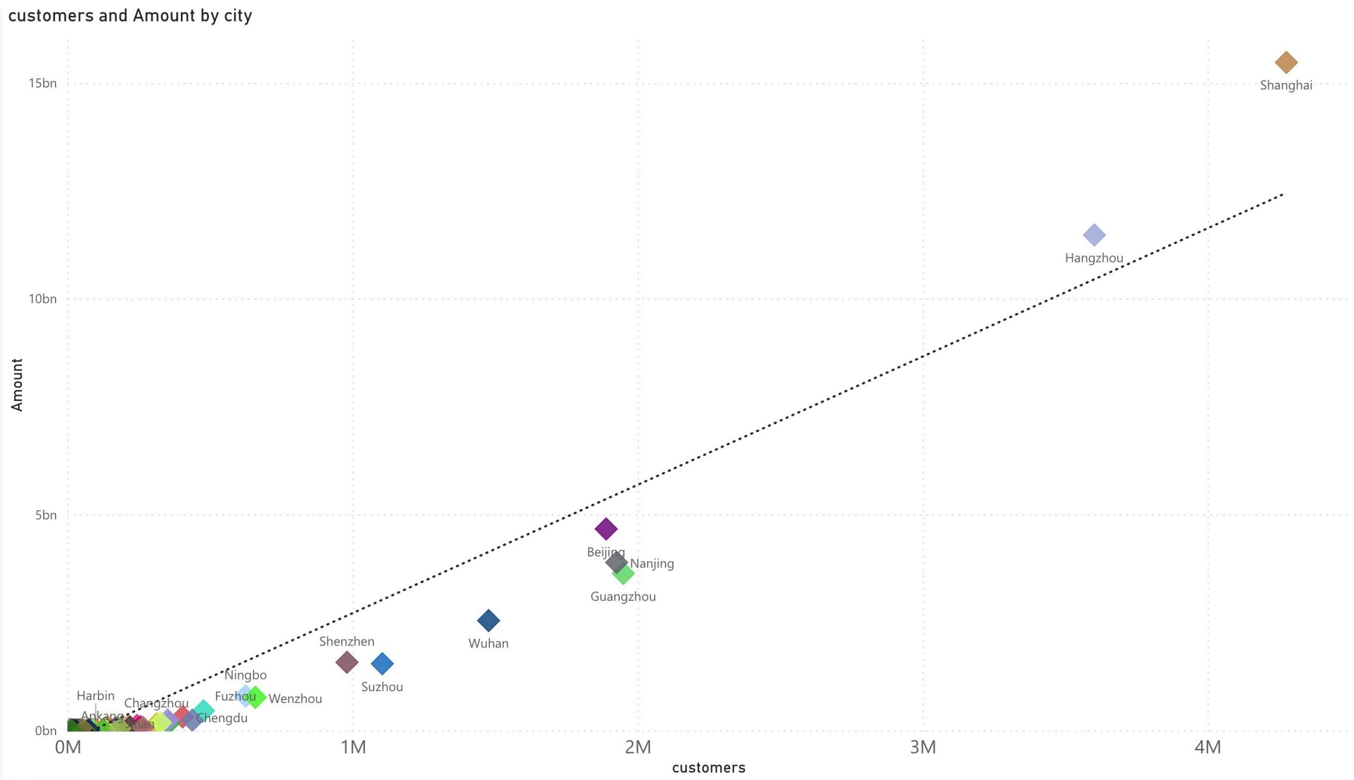
7.Customer Segmentation Analysis

7.1. Key Finding

This part of the report will discuss how to use customer's demographic, geography, behavior segmentation and RFM clustering based on exploratory data analysis for the stakeholders. In particualr, RFM clustering will give the recommendation as below, especially the frequency(F) and monetary(M) value can impact on customer's lifetime value, and recency(R) can impact on retention, such as:
1.When was the cusotmer last purchase was?
2.How often the cusotmer have purchasd in the recent period.
3.And how much the customer spent overall.

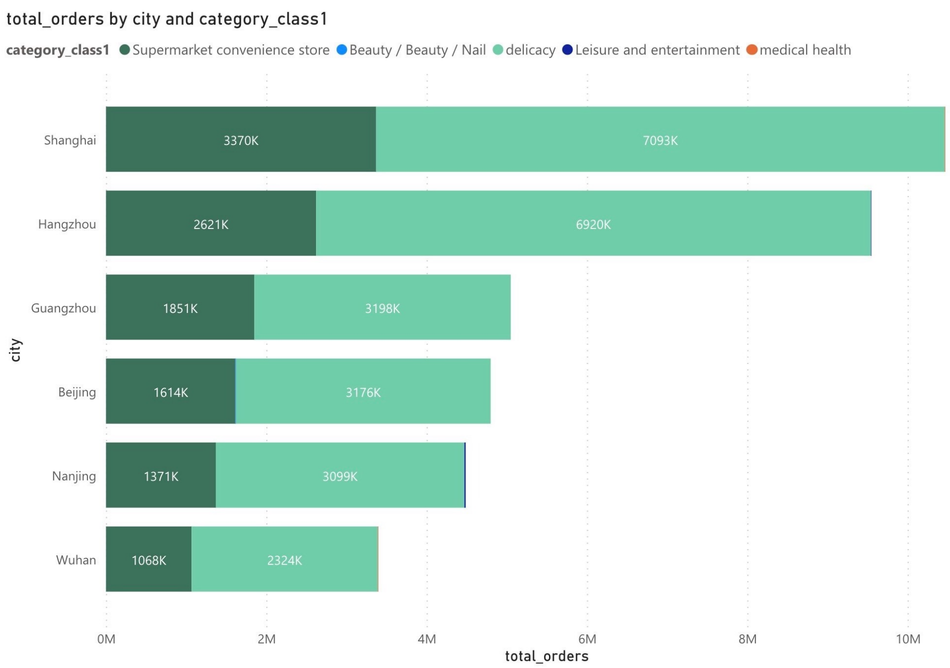
The customers in the top 6 cities have the most order amount

- The scatter chart screens the top 6 cities with the most amount of orders, they are :Shanghai,Hangzhou,Guangzhou,Beijing, Nanjing,Wuhan.



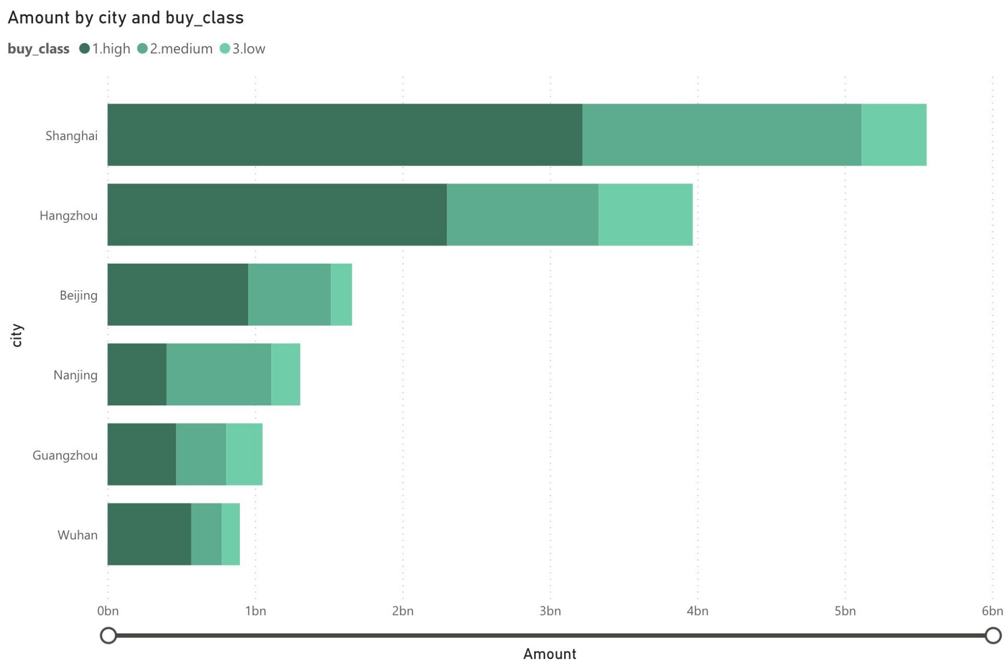
The customers having the most consumption levels in the top 6 cities

- The bar chart screens out the top 6 cities who has the most consumption level, they are shanghai,Hangzhou,Guangzhou,Beijing,Nanjing, and Wuhan.
- Totol customers for the 6 citys is 10802783, which is 54.88% of the total customers.
- Total orders for the 6 citys is 37751089, which is 54.18% of the total orders.



- For the six cities with top consumption level, dividing the segment of customers into high, medium and low of consumption level in each city。
- For the top 6 cities, customer segmetation should be considered according to 18(6*3) groups of customers due to the considerable number of customers with the various consumption levels.

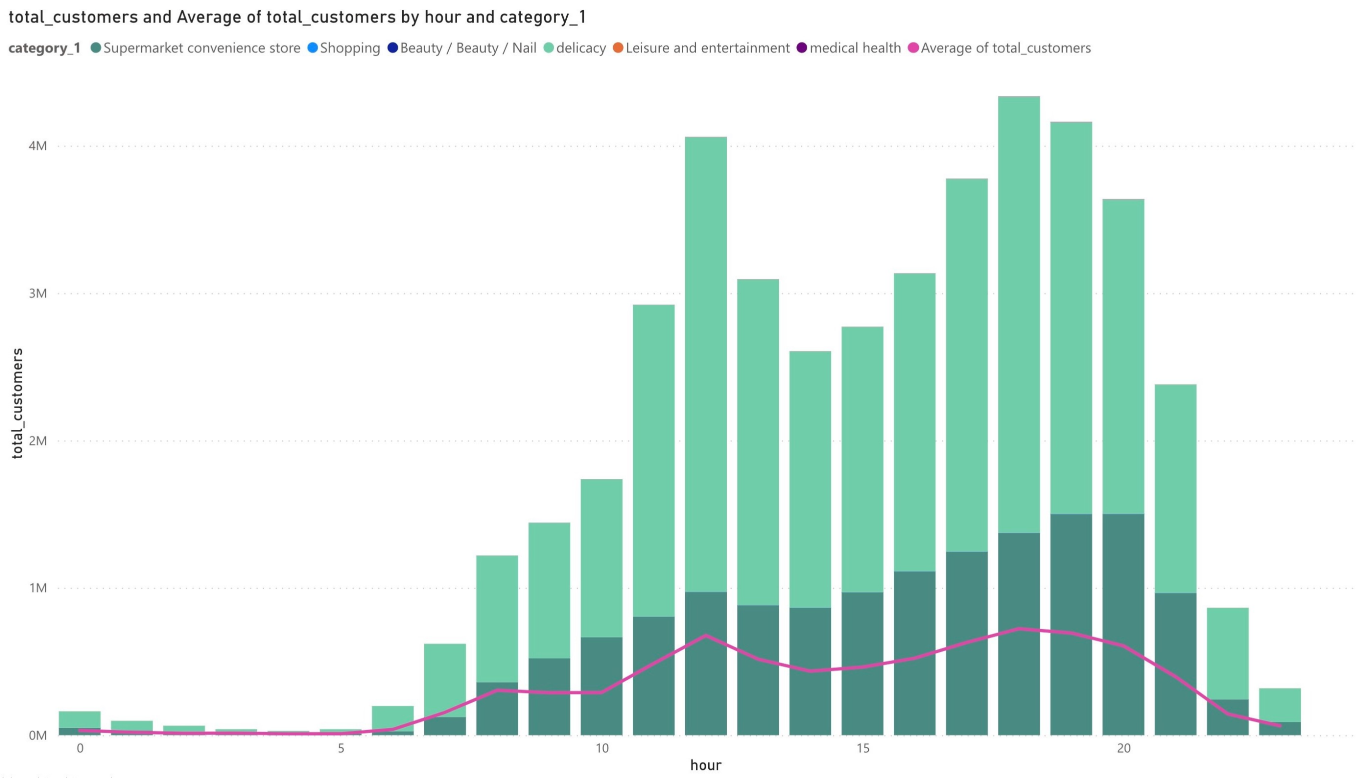
In other words, the six cities' customers were divided into high, medium and low consumption levels, then the geographic segmentation can be regarded as 18 groups of customers (6 cities*3 consumption levels).



Count of customers by product categories and hour

The bar chart screened out the customers who ordered products in the most popular category during a day:

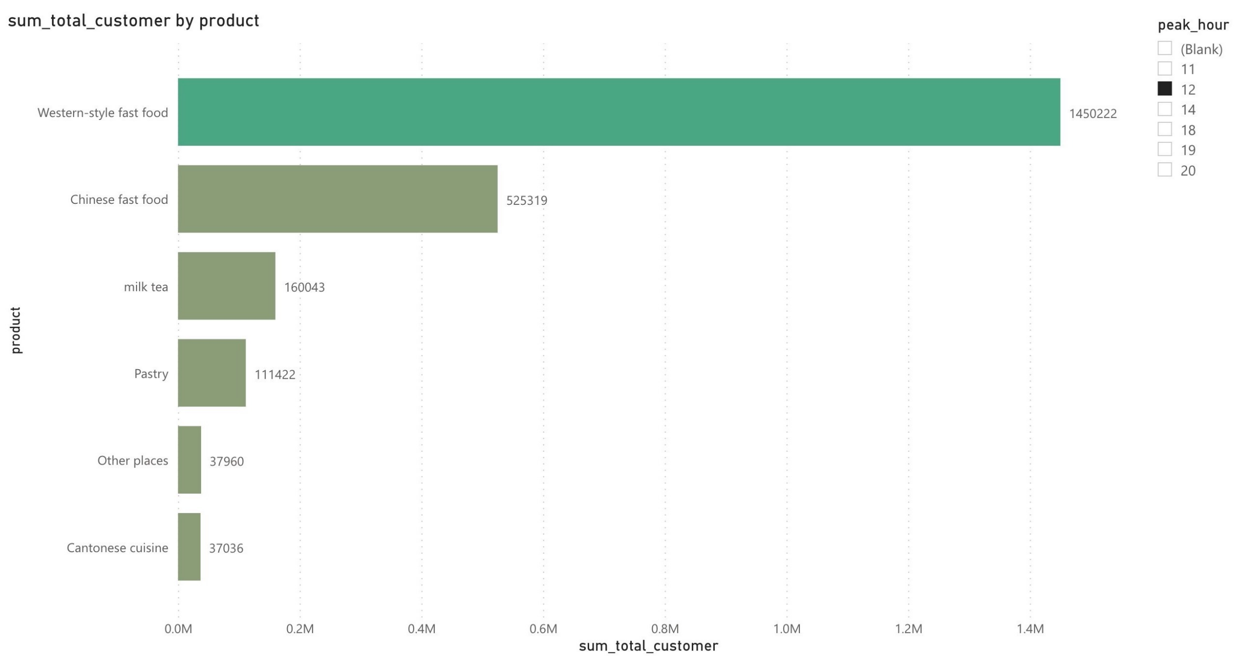
- The peak time for category of delicacy appears at 18:00.
- The peak time for Supermarket&Convenience store appears at 19:00~20:00.
- Adjusting marketing cost and executing periodically relevant campaigns that would boost sales during these peak times.



Count of customers by product categories during the peak hour

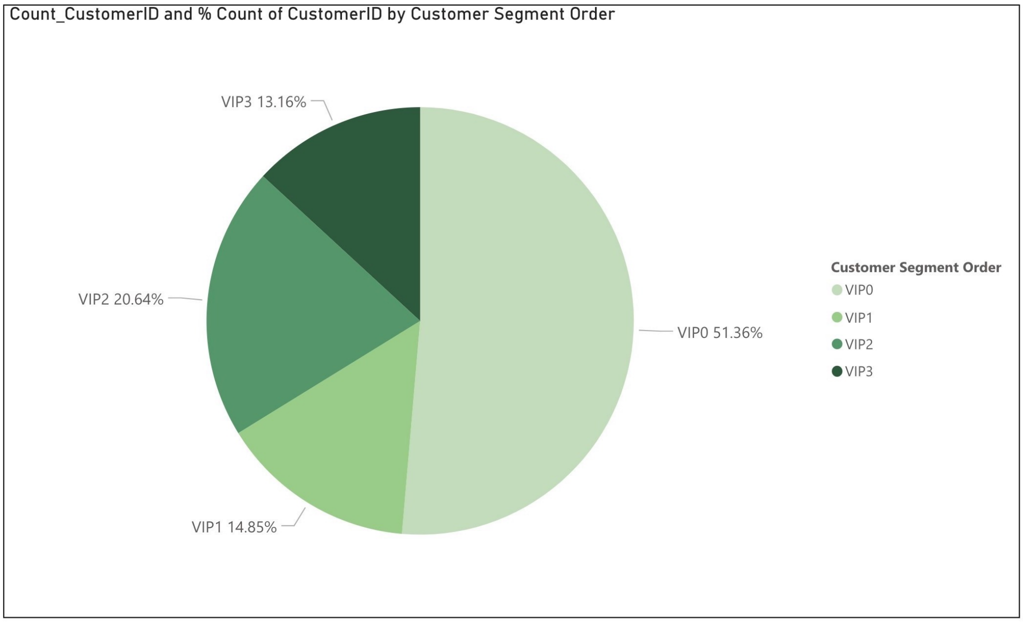
The bar chart screened out the most customers who ordered the most popular products during the peak hour:

- The most popular food at 12PM is Western-style fastfood(39.93%), then chinese fast food(14.46%).
- Adjusting marketing cost and executing periodically relevant campaigns that would boost sales during these peak times.



The customers categoried by RFM segmetatiom

- The metrics R could not have a obvious effect on RFM analysis, only F and M are about to considered as the determining metrics.The details of Recency analysis could refer to the part '8.2 Other analysis' for "Hero Customers" of this report.
- According to customer's values in the two dimensions of order frequency and order monetary, the customers are divided into four types: VIP3,VIP2,VIP1,VIP0
- The group of customers in quadrant VIP3 which both has frequency and monetary over average value, is more likely to convert the user's click action into actual purchase behavior.
- Totol customers of VIP3 is 1964775, which is 13% of the total customers.
- Total orders of VIP3 is 37751089, which is 54.18% of the total orders.
- The VIP3 group of customer is the most possible "Hero Customer".



Customer Segment Order	Count_CustomerID	% Count of CustomerID
VIP0	7670772	51.36
VIP2	3082246	20.64
VIP1	2217267	14.85
VIP3	1964775	13.16
Total	**14935060**	**100.00**

7.2 Other analysis for the Customers behavior

Date transformation

- Data transformation of datetime: partitioning pay_time dimension into fine granularities dimension

In [ ]:

```
%% sql

CREATE TABLE user_bh_p AS
WITH ub as(
    select *,to_timestamp(pay_time,'YYYY-MM-DD HH24:MI:SS') as datetime
    from user_bh_pay
)
select *
    ,date_part('year',datetime) as year
    ,date_part('quarter',datetime) as quarter
    ,date_part('month',datetime) as month
    ,date_part('week',datetime) as week
    ,date_part('day',datetime) as day
    ,date_part('hour',datetime)  as hour
from ub
```

- Data transformation of consumption level: Low, Medium, and High

In [ ]:

```
%%sql
CREATE TABLE shop_info as
SELECT  *
        ,case when perpay between 1 and 7 then  'low'
              when perpay between 8 and 12 then  'medium'
              else 'High'
              end as buy_class
from shopinfo
```

Screening the customers who have the most recent behavior of purchase

- Collected 1257,6771 of the orders record as below in the latest four months to calculate the R feature, the customers who had not appeared in the last four months of the records would be considered with churn instead of R.

In [ ]:

```
%%sql

with diff as (
    select
        user_id,
        max(datetime) as last_event,
        now()::date-max(datetime)::date as day_diff
    from user_bh_p
    where datetime>= timestamp'2016-06-22'
    group by user_id
    order by day_diff desc
), window_recency_top as(
    select user_id
    ,day_diff
    ,row_number() over (PARTITION by 1) rn
    from diff
)
select count(1) from window_recency_top
```

```
 * postgresql://postgres:***@this_postgres/postgres
1 rows affected.
```

Out[25]:

count
12576771

Then screened 265000 customers having the latest orders as R feature

- The analysis here used the algorithm to distribute and cluster customers equally into 6 groups based on R value.
- Since this historical data came from a certain period in the past, the difference between customers in R value was very small, so Recency was not considered as one of the dimensions of RFM feature as mentioned at first.
- The top 65000 customers had the most but the very similar order behavior of recency as the other customers, and the presentage of this customer segmentation is 0.09%.

In [6]:

```
%%sql

with diff as (
    select
        user_id,
        max(datetime) as last_event,
        now()::date-max(datetime)::date as day_diff
    from user_bh_p
    where datetime>= timestamp'2016-06-22'
    group by user_id
    order by day_diff desc
), window_recency_top as(
    select user_id
    ,last_event
    ,day_diff
    ,row_number() over (PARTITION by 1 order by day_diff) rn
    from diff
)
--select count(1) from window_recency_top--12576771
--select * from window_recency_top where rn in (1,2580000,4580000,6580000,8580000,10
select * from window_recency_top where rn in (1,400000,800000,1200000,1600000,200000
--select * from window_recency_top where rn in (1,65000,130000,195000,265000,330000,
--select * from window_recency_top where rn in (1,10000,20000,30000,40000,50000,6000
```

```
 * postgresql://postgres:***@this_postgres/postgres
7 rows affected.
```

Out[6]:

user_id	last_event	day_diff	rn
20073784	2016-10-31 13:00:00+00:00	1805	1
8014266	2016-10-30 18:00:00+00:00	1806	400000
18610116	2016-10-28 13:00:00+00:00	1808	800000
2069082	2016-10-26 13:00:00+00:00	1810	1200000
19380458	2016-10-24 14:00:00+00:00	1812	1600000
9297834	2016-10-22 13:00:00+00:00	1814	2000000
13862664	2016-10-18 19:00:00+00:00	1818	2580000

Screening top customers who have the most frenquency of purchase behavior as Frequency feature

- Collected 848,1514 of the orders record whose time span between first order and last order as below are over two weeks, then calculated the F feature.
- The analysis here used the algorithm to distribute and cluster customers equally among 6 groups based on the F feature.
- The top 28000 customers had the most order frequency, and the presentage of this customer segmentation is 0.14%.

In [15]:

```
%%sql
with grouped as(
    select user_id
    , count(btype) as count_buy_f
    , min(datetime) as first_event
    , max(datetime) as last_event
    , max(datetime)::date-min(datetime)::date as day_span
from user_bh_p
group by user_id
order by count_buy_f desc
), window_top_freq as (
    select user_id
    ,count_buy_f
    ,day_span
    ,(day_span/count_buy_f) as avg_day_span_per_order
    ,row_number() over (PARTITION by 1 order by count_buy_f desc) rn
    from grouped
    where  day_span>=14
)
--select count(1) from window_top_freq--8481514
--select * from window_top_freq where rn in (1,1413600,2827200,4240800,5654400,70680
--select * from window_top_freq where rn in (1,235600,471200,706800,942400,1178000,1
--select * from window_top_freq where rn in (1,40000,80000,120000,160000,200000,2356
select * from window_top_freq where rn in (1,7000,14000,21000,28000,35000,40000)
```

```
 * postgresql://postgres:***@this_postgres/postgres
7 rows affected.
```

Out[15]:

user_id	count_buy_f	day_span	avg_day_span_per_order	rn
20476580	299	179	0	1
1552426	122	468	3	7000
2955350	97	328	3	14000
3127201	83	288	3	21000
15728560	75	335	4	28000
12046524	69	382	5	35000
14763293	65	395	6	40000

Screening top numbers of customers who have the highest consumption level as Monetary feature

- Collected 2626,5463 of the orders record to calculate the M feature.

In [9]:

```
%%sql
with grouped as(
    select u.user_id
        ,row_number() over (PARTITION by 1 order by perpay) rn
        ,count(1)
    from user_bh_p u
    inner join shop_info s on u.shop_id=s.shopid
    group by 1,2
    order by amount desc
)
select count(1) from grouped
```

```
 * postgresql://postgres:***@this_postgres/postgres
1 rows affected.
```

Out[9]:

count
26265463

- The analysis here used the algorithm to distribute and cluster customers equally among 6 groups based on the M feature.
- The top 20000 had the most order amount, and the presentage of this customer segmentation is 0.41%.

In [27]:

```
%%sql
with grouped as(
    select u.user_id as uid
        ,s.perpay as amount
        ,count(1) as cnt_order
        ,s.perpay*count(1) as total_amount
    from user_bh_p u
    inner join shop_info s on u.shop_id=s.shopid
    group by 1,2
    order by total_amount desc
),windowed_top_m as(
    select uid
    ,total_amount
    ,cnt_order
    ,row_number() over (PARTITION by 1 order by total_amount desc) rn
    from grouped
)
--select count(1) from windowed_top_m--26265463
--select * from windowed_top_m where rn in (1,4000000,8000000,12000000,18000000,2200
--select * from windowed_top_m where rn in (1,650000,1300000,1950000,2600000,3250000
--select * from windowed_top_m where rn in (1,110000,220000,330000,440000,550000,650
select * from windowed_top_m where rn in (1,20000,40000,60000,80000,110000)
```

```
 * postgresql://postgres:***@this_postgres/postgres
6 rows affected.
```

Out[27]:

uid	total_amount	cnt_order	rn
9785313	5860	293	1
10342456	972	54	20000
17709950	731	43	40000
746829	612	204	60000
9995691	536	67	80000
11797334	459	27	110000

The customers orders distribution by consumption level, product categorys , locations

Creat a integral table named 'master_table' with all basic feature prepared to analysis

In [30]:

```
%%sql
with master_table as(
        SELECT u.*
            ,s.city_name
            ,s.perpay
            ,s.cate_1
            ,s.cate_2
            ,s.cate_3
            ,s.buy_class
      from user_bh_p u
      inner join shop_info s on u.shop_id=s.shopid
)
select   cate_1 as --cate_class_1
        ,cate_2 as --cate_class_2
        ,cate_3 as --cate_class_3
        ,count(1) as totaL_cate_orders
        ,sum(case when buy_class='low' then 1 else 0 end) as cnt_paylevel_low
        ,sum(case when buy_class='medium' then 1 else 0 end) as cnt_paylevel_medium
        ,sum(case when buy_class='high' then 1 else 0 end) as cnt_paylevel_high
from master_table
group by 1,2,3
order by 4 desc
limit 5
```

```
 * postgresql://postgres:***@this_postgres/postgres
5 rows affected.
```

Out[30]:

cate_1	cate_2	cate_3	total_cate_orders	cnt_paylevel_low	cnt_paylevel_medium	cnt
delicacies	fast food	western-style fast food	20236931	815963	8804966	
supermarket convenience store	supermarket	None	18933693	608534	1137455	
supermarket convenience store	convenience store	None	5803642	5620461	170444	
delicacies	fast food	chinese fast food	5625374	3429633	1547813	
delicacies	casual food	fresh fruit	2966309	1311001	946192	

Total orders, total customers and consumption level distributions by citys:

In [7]:

```
%%sql
with master_table AS(
        SELECT u.*
            ,s.city_name
            ,s.perpay
            ,s.cate_1
            ,s.cate_2
            ,s.cate_3
            ,s.buy_class
      from user_bh_p u
      inner join shop_info s on u.shop_id=s.shopid
)
select city_name
        ,perpay
        ,cate_1
        ,cate_2
        ,cate_3
        ,count(distinct user_id) total_users
        ,count(1)  total_orders
        ,count(1)*perpay as total_amount
        from master_table
        group by city_name, perpay,cate_1,cate_2,cate_3
        order by total_amount desc, total_users desc, total_orders desc
        limit 5
```

```
 * postgresql://postgres:***@this_postgres/postgres
5 rows affected.
```

Out[7]:

city_name	perpay	cate_1	cate_2	cate_3	total_users	total_orders	total_amount
shanghai	19	supermarket convenience store	supermarket	None	288484	997185	18946515
hangzhou	19	supermarket convenience store	supermarket	None	216890	860797	16355143
suzhou	20	supermarket convenience store	supermarket	None	222998	717351	14347020
shanghai	18	supermarket convenience store	supermarket	None	221822	568507	10233126
beijing	19	supermarket convenience store	supermarket	None	138654	525416	9982904

7.3 Further Considerations

- Customer engagement and consumer realization play an important role during e-commerce process, which can usually count on the customer segmentation. And thanks to various type of customer semantation method, the business and marketing goal can be clarified accurately and effiently. Consequently, customer segemtation before making business decision can be highly taken into consideration.

- The more refined the customer segmentation, the higher the customer's conversion rate. Specifically RFM segmentation is as a helpful mode, also can segment the marketing at the same cost.
- If we have a specific user traffic budget for marketing, we could turn the target customers into our consumers through customer's segmentation analysis, instead of randomly sending ads to anyone without high marketing conversion rates.

8. Main challenge

```
- Cleaned and uploaded large dataset from sqlite3 to postgres
- Sampled the small typical dataset to analysis, please refer to data sample files:
https://github.com/PengGitt/Customer_Segmentation/tree/main/data%20sample
```

8.1 Splited the large dataset into chunks and uploaded to postgres

In [9]:

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [10]:

```
%sql postgresql://postgres:password@this_postgres/postgres
```

In [2]:

```
from sqlalchemy import create_engine
import sqlite3
import pandas as pd
import csv
from pandasql import sqldf
from datetime import datetime
```

In [20]:

```
sq= sqlite3.connect('userbehavior.sqlite3')
pg= create_engine('postgresql://postgres:password@this_postgres')
```

In [5]:

```
%%sql
show database
```

```
 * postgresql://postgres:***@this_postgres/postgres
(psycopg2.errors.UndefinedObject) unrecognized configuration parameter
"database"

[SQL: show database]
(Background on this error at: https://sqlalche.me/e/14/f405) (https://
sqlalche.me/e/14/f405))
```

In [ ]:

```
sql="Select *, 'buy' as btype from userpay"
for df in pd.read_sql(sql,sq,chunksize=200000):
    df.to_sql('user_bh_pay',pg,if_exists='append')
    print('loaded more 200000 rows')
```

loaded more 200000 rows
loaded more 200000 rows
loaded more 200000 rows

8.2 Data overview and data cleaning

Overview

In [3]:

```
%%sql
select count(1) total_order
     , count(distinct user_id) total_user
     , count(distinct shop_id) cnt_product_category from user_bh_p
```

```
 * postgresql://postgres:***@this_postgres/postgres
1 rows affected.
```

Out[3]:

total_order	total_user	cnt_product_category
69674110	19583949	2000

Data cleaning:

```
  - Relationships check: ship_id check(foreign key)
  - Not-null check
  - Accepted values: shopid(1-2000),perpay(1-20)
```

- **Relationship check: 0 of result is correct for the relationship**

In [11]:

```
%%sql
select count(distinct shop_id) as out_of_foreign from user_bh_p where shop_id not in
```

```
 * postgresql://postgres:***@this_postgres/postgres
1 rows affected.
```

Out[11]:

out_of_foreign
0

- **Not-null check: The count results of the three fields are the same**

In [15]:

```
%%sql
select count(1),count(pay_time),count(shop_id) from user_bh_p
```

```
 * postgresql://postgres:***@this_postgres/postgres
1 rows affected.
```

Out[15]:

count	count_1	count_2
69674110	69674110	69674110

- **Accepted Values Check: 0 of result is correct for the accepted value**

In [18]:

```
%%sql
select count(distinct shopid) as cnt_out_of_shopid from shop_info where shopid not b
```

```
 * postgresql://postgres:***@this_postgres/postgres
1 rows affected.
```

Out[18]:

cnt_out_of_shopid
0

In [19]:

```
%%sql
select count(distinct perpay) as cnt_out_of_perpay from shop_info where perpay not b
```

```
 * postgresql://postgres:***@this_postgres/postgres
1 rows affected.
```

Out[19]:

cnt_out_of_perpay
0

8.3 Understanding user sample groups from the large data set

Extracted top 10 customers with the most orders as sample

In [39]:

```
#sq=sqlite3.connect('userbehavior.sqlite3')
#top_10=pd.read_sql('select user_id, count(1) as cnt_total_order from userpay group
#top_10.to_csv('top10_user.csv') #find top10 user_id by cnt

top10_user = pd.read_csv('top10_user.csv',index_col=0)
top10_user
```

Out[39]:

	user_id	cnt_total_order
0	20476580	299
1	2716941	297
2	16549240	296
3	19677677	295
4	6712547	295
5	5972671	295
6	21649568	294
7	21586973	294
8	17739226	294
9	3450024	294

Transfromed the sample data by dimension(category,order time,etc...)

In [31]:

```
# sq=sqlite3.connect('userbehavior.sqlite3')
# top_10_behavior=pd.read_sql(
#     'select * from userpay where user_id in(20476580,2716941,16549240,19677677,671
# top_10_behavior.to_csv('top10_userbehavior.csv')
top10_userbehavior = pd.read_csv('top10_userbehavior.csv',index_col=0)
top10_userbehavior.head()
```

Out[31]:

	user_id	shop_id	pay_time
0	17739226	1302	2016-07-11 10:00:00
1	17739226	1302	2016-06-11 16:00:00
2	17739226	1302	2016-06-09 16:00:00
3	17739226	1302	2016-05-22 22:00:00
4	17739226	1302	2016-08-20 12:00:00

Loaded the sample user data feature to be ready for analysis

In [37]:

```
data=pd.read_csv('top10_userbehavior.csv',index_col=0)

def  run_sql(sql:str) -> pd.DataFrame:
    _df=sqldf(sql)
    return _df

user_feature=run_sql('''
   --begin-sql
   select
    user_id
     ,COUNT(1) as count_order
     ,COUNT(distinct shop_id) as count_item
     ,DATE(min(pay_time)) as first_ordertime
     ,Date(max(pay_time)) as last_ordertime
     ,CAST(julianday(date(max(pay_time)))-julianday(date(min(pay_time))) as INT) as

 from data
 group by 1
 order by 6 desc
 --end-sql
''')

user_feature
```

Out[37]:

	user_id	count_order	count_item	first_ordertime	last_ordertime	days_on_platform
0	6712547	295	2	2015-06-29	2016-10-30	489
1	21586973	294	1	2015-11-19	2016-10-31	347
2	16549240	296	1	2015-11-18	2016-10-29	346
3	2716941	297	2	2015-11-18	2016-10-28	345
4	3450024	294	2	2015-12-04	2016-10-28	329
5	5972671	295	3	2015-11-18	2016-10-02	319
6	17739226	294	1	2015-11-29	2016-10-04	310
7	21649568	294	1	2015-11-17	2016-07-14	240
8	19677677	295	1	2015-11-17	2016-07-13	239
9	20476580	299	3	2016-03-06	2016-09-01	179

9. Disclaimer

The sole purpose of this research is to provide as many features as possible for customers segementation to alibaba's merchants.