京东订单数据分析

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from matplotlib.ticker import FuncFormatter
plt.rcParams['font.sans-serif']=['Arial Unicode MS']

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
order = 'course_order_d.csv'
df = pd.read_csv(order,sep='\t', encoding="utf-8", dtype=str)
```

In [3]:

```
df.head()
```

Out[3]:

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	item_sku
0	linfbi007	116828823929	116828823929	2020-05-25 18:09:39.0	2020-05-25	100000350
1	13601089905_p	116769479986	121562216719	2020-05-25 00:04:15.0	2020-05-25	100000350
2	jd_UbSjKwFGOfbv	116815391384	116809219025	2020-05-25 13:47:33.0	2020-05-25	100000350
3	yangwangjun1300	116814673181	116814673181	2020-05-25 14:34:25.0	2020-05-25	100000350
4	jd_77dbadc203044	116811074034	116811074034	2020-05-25 14:47:42.0	2020-05-25	100000350

5 rows × 23 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76631 entries, 0 to 76630
Data columns (total 23 columns):
user_log_acct              76631 non-null object
parent_sale_ord_id         76631 non-null object
sale_ord_id                76631 non-null object
sale_ord_tm                76631 non-null object
sale_ord_dt                76631 non-null object
item_sku_id                76631 non-null object
item_name                  76631 non-null object
brandname                  76631 non-null object
sale_qtty                  76631 non-null object
item_first_cate_name       76631 non-null object
item_second_cate_name      76631 non-null object
item_third_cate_name       76631 non-null object
before_prefr_unit_price    76631 non-null object
after_prefr_unit_price     76631 non-null object
user_actual_pay_amount     76631 non-null object
sale_ord_valid_flag        76631 non-null object
cancel_flag                76631 non-null object
check_account_tm           53360 non-null object
total_offer_amount         76631 non-null object
self_ord_flag              76631 non-null object
user_site_city_id          38441 non-null object
user_site_province_id      38598 non-null object
user_lv_cd                 76631 non-null object
dtypes: object(23)
memory usage: 13.4+ MB
```

In [5]:

```
df.isnull().sum().sort_values(ascending=False)
```

Out[5]:

```
user_site_city_id          38190
user_site_province_id      38033
check_account_tm           23271
user_lv_cd                     0
item_first_cate_name           0
parent_sale_ord_id             0
sale_ord_id                    0
sale_ord_tm                    0
sale_ord_dt                    0
item_sku_id                    0
item_name                      0
brandname                      0
sale_qtty                      0
item_third_cate_name           0
item_second_cate_name          0
before_prefr_unit_price        0
after_prefr_unit_price         0
user_actual_pay_amount         0
sale_ord_valid_flag            0
cancel_flag                    0
total_offer_amount             0
self_ord_flag                  0
user_log_acct                  0
dtype: int64
```

In [6]:

```
df['sale_ord_dt'].unique()
```

Out[6]:

```
array(['2020-05-25'], dtype=object)
```

In [7]:

```
df['sale_qtty'] = df['sale_qtty'].astype('int')
df['sale_ord_valid_flag'] = df['sale_ord_valid_flag'].astype('int')
df['cancel_flag'] = df['cancel_flag'].astype('int')
df['self_ord_flag'] = df['self_ord_flag'].astype('int')
```

In [8]:

```
df['before_prefr_unit_price'] = df['before_prefr_unit_price'].astype('float')
df['after_prefr_unit_price'] = df['after_prefr_unit_price'].astype('float')
df['user_actual_pay_amount'] = df['user_actual_pay_amount'].astype('float')
df['total_offer_amount'] = df['total_offer_amount'].astype('float')
```

In [9]:

```
df.loc[:,'check_account_tm '] = pd.to_datetime(df.loc[:,'check_account_tm'])
df.loc[:,'sale_ord_tm'] = pd.to_datetime(df.loc[:,'sale_ord_tm'])
df.loc[:,'sale_ord_dt'] = pd.to_datetime(df.loc[:,'sale_ord_dt'])
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76631 entries, 0 to 76630
Data columns (total 24 columns):
user_log_acct              76631 non-null object
parent_sale_ord_id         76631 non-null object
sale_ord_id                76631 non-null object
sale_ord_tm                76631 non-null datetime64[ns]
sale_ord_dt                76631 non-null datetime64[ns]
item_sku_id                76631 non-null object
item_name                  76631 non-null object
brandname                  76631 non-null object
sale_qtty                  76631 non-null int32
item_first_cate_name       76631 non-null object
item_second_cate_name      76631 non-null object
item_third_cate_name       76631 non-null object
before_prefr_unit_price    76631 non-null float64
after_prefr_unit_price     76631 non-null float64
user_actual_pay_amount     76631 non-null float64
sale_ord_valid_flag        76631 non-null int32
cancel_flag                76631 non-null int32
check_account_tm           53360 non-null object
total_offer_amount         76631 non-null float64
self_ord_flag              76631 non-null int32
user_site_city_id          38441 non-null object
user_site_province_id      38598 non-null object
user_lv_cd                 76631 non-null object
check_account_tm           53360 non-null datetime64[ns]
dtypes: datetime64[ns](3), float64(4), int32(4), object(13)
memory usage: 12.9+ MB
```

缺失值 & 异常值 处理

In [11]:

```
(df.loc[:,'before_prefr_unit_price']<288).sum() # 优惠前冰箱的最低价格为288元，低于此价格的订单认为是异常订单
```

Out[11]:

14252

In [12]:

```
(df.loc[:,'after_prefr_unit_price']<0).sum()
```

Out[12]:

0

In [13]:

```
(df.loc[:,'user_actual_pay_amount']<0).sum()
```

Out[13]:

0

In [14]:

```
(df.loc[:,'total_offer_amount']<0).sum()
```

Out[14]:

0

In [15]:

```
df = df[df['before_prefr_unit_price' ]>= 288]
print('删除异常值后：',df.shape)
```

删除异常值后： (62379, 24)

In [16]:

```
df.sale_ord_id.duplicated()
```

Out[16]:

```
0        False
1        False
2        False
3        False
4        False
5        False
6        False
7        False
8        False
9        False
10       False
11       False
12       False
13       False
14       False
15       False
16       False
17       False
18       False
19       False
20       False
21       False
22       False
23       False
24       False
25       False
26       False
27       False
28       False
29       False
         ...
76601    False
76602    False
76603     True
76604    False
76605    False
76606    False
76607    False
76608    False
76609    False
76610    False
76611    False
76612    False
76613    False
76614    False
76615    False
76616    False
76617    False
76618    False
76619    False
76620     True
76621    False
76622    False
76623     True
76624    False
76625    False
76626    False
76627    False
76628    False
```

```
76629    False
76630    False
Name: sale_ord_id, Length: 62379, dtype: bool
```

In [17]:

```
df.drop_duplicates(subset=['sale_ord_id'],keep='first',inplace=True) # 去掉订单号重复的数据（这里京东的建议保留第一个）
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 61536 entries, 0 to 76630
Data columns (total 24 columns):
user_log_acct              61536 non-null object
parent_sale_ord_id         61536 non-null object
sale_ord_id                61536 non-null object
sale_ord_tm                61536 non-null datetime64[ns]
sale_ord_dt                61536 non-null datetime64[ns]
item_sku_id                61536 non-null object
item_name                  61536 non-null object
brandname                  61536 non-null object
sale_qtty                  61536 non-null int32
item_first_cate_name       61536 non-null object
item_second_cate_name      61536 non-null object
item_third_cate_name       61536 non-null object
before_prefr_unit_price    61536 non-null float64
after_prefr_unit_price     61536 non-null float64
user_actual_pay_amount     61536 non-null float64
sale_ord_valid_flag        61536 non-null int32
cancel_flag                61536 non-null int32
check_account_tm           41611 non-null object
total_offer_amount         61536 non-null float64
self_ord_flag              61536 non-null int32
user_site_city_id          31585 non-null object
user_site_province_id      31739 non-null object
user_lv_cd                 61536 non-null object
check_account_tm           41611 non-null datetime64[ns]
dtypes: datetime64[ns](3), float64(4), int32(4), object(13)
memory usage: 10.8+ MB
```

In [18]:

```
df.user_site_city_id=df.user_site_city_id.fillna('Not Given')
df.user_site_province_id =df.user_site_province_id.fillna('Not Given')
```

In [19]:

```
df.describe()
```

Out[19]:

	sale_qtty	before_prefr_unit_price	after_prefr_unit_price	user_actual_pay_amount
count	61536.000000	61536.000000	61536.000000	61536.000000
mean	1.105158	2197.308403	1904.257668	853.219319
std	1.257971	1802.808343	1701.518805	1414.997061
min	1.000000	288.000000	1.000000	0.000000
25%	1.000000	919.000000	729.000000	0.000000
50%	1.000000	1499.000000	1299.000000	325.990000
75%	1.000000	3299.000000	2699.000000	1196.937500
max	80.000000	21999.000000	21999.000000	87920.000000

In [20]:

```
df['total_actual_pay'] = df['sale_qtty'] * df['after_prefr_unit_price']
df
```

Out[20]:

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
0	linfbi007	116828823929	116828823929	2020-05-25 18:09:39	2020-05-25	1000(
1	13601089905_p	116769479986	121562216719	2020-05-25 00:04:15	2020-05-25	1000(
2	jd_UbSjKwFGOfbv	116815391384	116809219025	2020-05-25 13:47:33	2020-05-25	1000(
3	yangwangjun1300	116814673181	116814673181	2020-05-25 14:34:25	2020-05-25	1000(
4	jd_77dbadc203044	116811074034	116811074034	2020-05-25 14:47:42	2020-05-25	1000(
5	葑存记忆	121591740399	121591740399	2020-05-25 11:38:12	2020-05-25	1000(
6	jd_uuwhuTZixluQ	116825666739	116825666739	2020-05-25 17:09:30	2020-05-25	1000(
7	haiqiang0307	116881662586	116881662586	2020-05-25 23:24:22	2020-05-25	1000(

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
8	shangchunping0216	116847011379	116847011379	2020-05-25 20:12:12	2020-05-25	1000(
9	jd_5d3fef6659091	116835354680	116835354680	2020-05-25 17:20:53	2020-05-25	1000(
10	9657401-591066	116810188022	116810188022	2020-05-25 14:17:39	2020-05-25	1000(
11	jd_5cbf61ca1dda1	121624638855	121624638855	2020-05-25 00:00:07	2020-05-25	1000(
12	maxiaohong1985	116785989012	116785989012	2020-05-25 08:22:11	2020-05-25	1000(
13	72027942-163865	116851627964	116851627964	2020-05-25 20:24:41	2020-05-25	1000(
14	ice_呗	116816666996	116816666996	2020-05-25 15:17:53	2020-05-25	1000(
15	jd_4452c13c382f7	116836284477	116836284477	2020-05-25 18:09:11	2020-05-25	1000(

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
16	jd_76b772039a54e	121623032619	121623032619	2020-05-25 00:01:56	2020-05-25	1000(
17	287122990_m	116767234713	116767234713	2020-05-25 00:09:49	2020-05-25	1000(
18	jd_67eda241885a8	116775076123	116775076123	2020-05-25 10:03:11	2020-05-25	1000(
19	mxd_816	116795056082	116795056082	2020-05-25 11:26:17	2020-05-25	1000(
20	posidon001	116790873818	116790873818	2020-05-25 08:32:01	2020-05-25	1000(
21	wlx1993	116803837055	116803837055	2020-05-25 12:38:17	2020-05-25	1000(
22	924288726_m	116827329014	116827329014	2020-05-25 17:18:08	2020-05-25	1000(
23	TracyJordan123	116791411932	121647436968	2020-05-25 10:15:06	2020-05-25	1000(

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
24	jd_6569c2782e2e9	116830314520	116830314968	2020-05-25 16:33:37	2020-05-25	1000(
25	suihjbuy	116866820625	116866820625	2020-05-25 22:13:09	2020-05-25	1000(
26	116532220-413441	116798948087	116798948087	2020-05-25 11:55:30	2020-05-25	1000(
27	287122990_m	116810099005	116810099005	2020-05-25 13:38:34	2020-05-25	1000(
28	jd_4ab09a8c03310	116878273919	121593736079	2020-05-25 23:29:19	2020-05-25	1000(
29	924288726_m	116827329014	116829399068	2020-05-25 17:18:08	2020-05-25	1000(
...	...	...	...	...	...	
76598	jd_wggqMlNwcKZu	116828835568	116828729619	2020-05-25 17:37:01	2020-05-25	
76599	jd_45370c1b9c0fc	116805964089	116805964089	2020-05-25 14:18:12	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76600	jd_67e7edfc9ebdc	116808387729	116808387729	2020-05-25 13:58:41	2020-05-25	
76601	jd_696d1396be5b1	116798619729	116798619729	2020-05-25 12:05:12	2020-05-25	
76602	红樱桃JD	116864510111	116864510111	2020-05-25 21:43:33	2020-05-25	
76604	jd_SIuXCfEaqtcC	116849725400	116843295444	2020-05-25 19:31:07	2020-05-25	
76605	18608085791_p	121626875846	121626875846	2020-05-25 00:15:05	2020-05-25	
76606	jd_bkoRKZuCoxCc	116839777944	116839777944	2020-05-25 17:59:06	2020-05-25	
76607	jd_YdkGakfRdRGp	116839288691	116839295152	2020-05-25 19:11:16	2020-05-25	
76608	215414ZMD001	121562570798	121562570798	2020-05-25 00:04:22	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76609	aleeyoung	116825271571	116825271571	2020-05-25 17:05:12	2020-05-25	
76610	18933954534_p	116837089872	116837089872	2020-05-25 18:49:49	2020-05-25	
76611	jd_69424511b4ac2	116817061872	116817061872	2020-05-25 15:43:53	2020-05-25	
76612	513028ZMD000	121563801487	121563801487	2020-05-25 00:25:04	2020-05-25	
76613	jd_688188e631142	116833414224	116833414224	2020-05-25 18:17:48	2020-05-25	
76614	350127ZMD999	116814845655	116814845655	2020-05-25 15:03:17	2020-05-25	
76615	jd_DWeSFeKLXKrd	116839674581	116839674581	2020-05-25 19:01:30	2020-05-25	
76616	jd_ISfRFHnWvHTY	116824138585	116824138585	2020-05-25 17:29:04	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76617	jd_6091df073a467	116868694552	116868694552	2020-05-25 21:34:48	2020-05-25	
76618	jd_4209b7a5111f3	116878457077	116878457077	2020-05-25 23:46:19	2020-05-25	
76619	jd_xiIJnaUNQcbS	116841625265	116841857270	2020-05-25 19:29:46	2020-05-25	
76621	513401ZMD001	116786238803	116786238803	2020-05-25 09:33:51	2020-05-25	
76622	jd_uynMoFJyDDxL	116786509052	116786509052	2020-05-25 08:16:30	2020-05-25	
76624	jd_7b852d8fa7721	116798024469	116798024469	2020-05-25 11:40:35	2020-05-25	
76625	350583ZMD066	116804271579	116804271579	2020-05-25 15:40:45	2020-05-25	
76626	sd32513500	116849959579	116849959579	2020-05-25 21:48:53	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76627	jd_EHMnIsCGtVpC	116836914066	116825379579	2020-05-25 18:58:47	2020-05-25	
76628	jd_cssQCfiQSNbR	116829094940	116829094940	2020-05-25 17:15:40	2020-05-25	
76629	441900ZMD666	116841557298	116841557298	2020-05-25 19:40:55	2020-05-25	
76630	u_7adb4edaf1604	116795897298	116795897298	2020-05-25 11:46:39	2020-05-25	

61536 rows × 25 columns

宏观分析

In [21]:

```
#取消订单数量
order_cancel = df[df.cancel_flag==1]['sale_ord_id'].count()
order_cancel
```

Out[21]:

17782

In [22]:

```
#订单数量
order_num = df['sale_ord_id'].count()
order_num
```

Out[22]:

61536

In [23]:

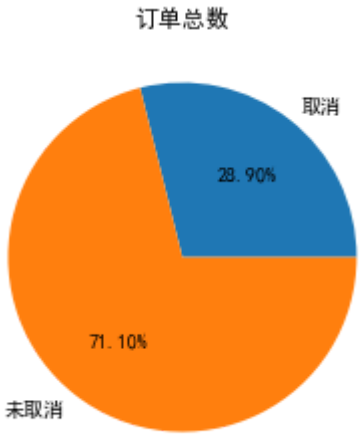
```
# 解决matplotlib中文乱码

matplotlib.rcParams['font.sans-serif'] = ['SimHei']
matplotlib.rcParams['font.serif'] = ['SimHei']
matplotlib.rcParams['axes.unicode_minus'] = False
```

In [24]:

```
labels = ['取消','未取消']
X = [order_cancel, order_num-order_cancel]
fig = plt.figure()
plt.pie(X,labels=labels,autopct='%1.2f%%') # autopct :控制饼图内百分比设置, '%1.1f'指小数点前后位数(没有用空格补齐);
plt.title("订单总数")
```

Out[24]:

Text(0.5, 1.0, '订单总数')



In [25]:

```
df2 = df.copy()
df2 = df2[(df2['sale_ord_valid_flag'] == 1)&(df2['cancel_flag'] == 0)&('before_prefr_unit_price'
 != 0)] # df2只包含有效订单
```

In [26]:

```
#有效订单数量
order_vaild = df2['sale_ord_id'].count()
order_vaild
```

Out[26]:

33846

In [27]:

```
#支付订单数量
order_payed = df2['sale_ord_id'][df2['user_actual_pay_amount'] != 0].count()
order_payed
```

Out[27]:

28769

In [28]:

```
#未支付订单数量
order_unpay = df2['sale_ord_id'][df2['user_actual_pay_amount'] == 0].count()
order_unpay
```

Out[28]:

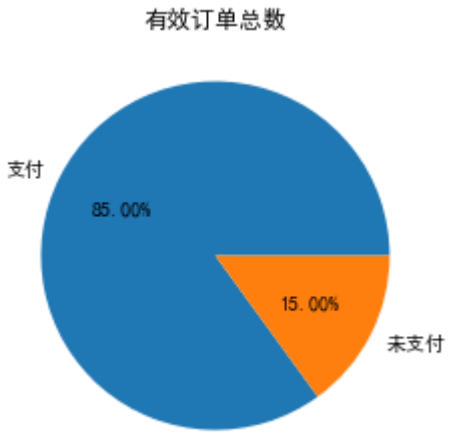
5077

In [29]:

```
labels = ['支付','未支付']
Y = [order_payed, order_unpay]
fig = plt.figure()
plt.pie(Y,labels=labels,autopct='%1.2f%%')
plt.title("有效订单总数")
```

Out[29]:

Text(0.5, 1.0, '有效订单总数')



订单的价格分布

In [30]:

```
price_series = df2['after_prefr_unit_price']
price_series
```

Out[30]:

```
0        1099.0
1        1099.0
2        1099.0
3        1099.0
4        1099.0
5        1099.0
6        1099.0
7        1099.0
8        1099.0
9        1099.0
10       1099.0
12       1099.0
16       1099.0
18       1099.0
19       1099.0
21       1099.0
23       1099.0
24       1099.0
25       1099.0
27       1099.0
28       1099.0
29       1099.0
30       1099.0
31       1099.0
33       1099.0
35       1099.0
39       1099.0
40       1099.0
42       1099.0
45       1099.0
          ...
76580     769.0
76581     769.0
76582     769.0
76584     769.0
76586     769.0
76587     769.0
76590     769.0
76592     769.0
76593     690.0
76596     769.0
76597     769.0
76598     769.0
76599     769.0
76601     769.0
76602     769.0
76604     769.0
76605     769.0
76608     690.0
76609     769.0
76611     769.0
76612     690.0
76613     769.0
76614     690.0
76617     769.0
76618     769.0
76621     999.0
76624     769.0
76625     690.0
```

```
76626     769.0
76629     690.0
Name: after_prefr_unit_price, Length: 33846, dtype: float64
```

In [31]:

```
price_series_num = price_series.count()

hist, bin_edges = np.histogram(price_series, bins=80) #生成直方图函数
hist_sum = np.cumsum(hist)
hist_per = hist_sum / price_series_num

print('hist:{}'.format(hist))
print('*'*100)
print('bin_edges:{}'.format(bin_edges))
print('*'*100)
print('hist_sum:{}'.format(hist_sum))
```

```
hist:[   3 1526 4719  817 3544 2705 3392  979 2078  430 1201 1099  905 1295
  189  896  912  866  832  235  849 1340  359  207  136  223  258   52
   40  235  121    5  221  118  303   96   51   52    0  161    0   20
    1   52    1    0   21    6    0    0  137    0    1   58    0    0
    4    1    2    2    0    2    0    0    0   24   45    0    3    0
    1    1    0    1    0    0    1    0    0   12]
****************************************************************************************
******************
bin_edges:[1.0000000e+00 1.5947500e+02 3.1795000e+02 4.7642500e+02 6.3490000e+02
 7.9337500e+02 9.5185000e+02 1.1103250e+03 1.2688000e+03 1.4272750e+03
 1.5857500e+03 1.7442250e+03 1.9027000e+03 2.0611750e+03 2.2196500e+03
 2.3781250e+03 2.5366000e+03 2.6950750e+03 2.8535500e+03 3.0120250e+03
 3.1705000e+03 3.3289750e+03 3.4874500e+03 3.6459250e+03 3.8044000e+03
 3.9628750e+03 4.1213500e+03 4.2798250e+03 4.4383000e+03 4.5967750e+03
 4.7552500e+03 4.9137250e+03 5.0722000e+03 5.2306750e+03 5.3891500e+03
 5.5476250e+03 5.7061000e+03 5.8645750e+03 6.0230500e+03 6.1815250e+03
 6.3400000e+03 6.4984750e+03 6.6569500e+03 6.8154250e+03 6.9739000e+03
 7.1323750e+03 7.2908500e+03 7.4493250e+03 7.6078000e+03 7.7662750e+03
 7.9247500e+03 8.0832250e+03 8.2417000e+03 8.4001750e+03 8.5586500e+03
 8.7171250e+03 8.8756000e+03 9.0340750e+03 9.1925500e+03 9.3510250e+03
 9.5095000e+03 9.6679750e+03 9.8264500e+03 9.9849250e+03 1.0143400e+04
 1.0301875e+04 1.0460350e+04 1.0618825e+04 1.0777300e+04 1.0935775e+04
 1.1094250e+04 1.1252725e+04 1.1411200e+04 1.1569675e+04 1.1728150e+04
 1.1886625e+04 1.2045100e+04 1.2203575e+04 1.2362050e+04 1.2520525e+04
 1.2679000e+04]
****************************************************************************************
******************
hist_sum:[    3  1529  6248  7065 10609 13314 16706 17685 19763 20193 21394 22493
 23398 24693 24882 25778 26690 27556 28388 28623 29472 30812 31171 31378
 31514 31737 31995 32047 32087 32322 32443 32448 32669 32787 33090 33186
 33237 33289 33289 33450 33450 33470 33471 33523 33524 33524 33545 33551
 33551 33551 33688 33688 33689 33747 33747 33747 33751 33752 33754 33756
 33756 33758 33758 33758 33758 33782 33827 33827 33830 33830 33831 33832
 33832 33833 33833 33833 33834 33834 33834 33846]
```

In [32]:

```
hist_per
```

Out[32]:

```
array([8.86367665e-05, 4.51752053e-02, 1.84600839e-01, 2.08739585e-01,
       3.13449152e-01, 3.93369970e-01, 4.93588607e-01, 5.22513739e-01,
       5.83909472e-01, 5.96614076e-01, 6.32098328e-01, 6.64568930e-01,
       6.91307688e-01, 7.29569225e-01, 7.35153342e-01, 7.61626189e-01,
       7.88571766e-01, 8.14158246e-01, 8.38740176e-01, 8.45683389e-01,
       8.70767594e-01, 9.10358683e-01, 9.20965550e-01, 9.27081487e-01,
       9.31099687e-01, 9.37688353e-01, 9.45311115e-01, 9.46847486e-01,
       9.48029309e-01, 9.54972523e-01, 9.58547539e-01, 9.58695267e-01,
       9.65224842e-01, 9.68711221e-01, 9.77663535e-01, 9.80499911e-01,
       9.82006736e-01, 9.83543107e-01, 9.83543107e-01, 9.88299947e-01,
       9.88299947e-01, 9.88890859e-01, 9.88920404e-01, 9.90456775e-01,
       9.90486320e-01, 9.90486320e-01, 9.91106778e-01, 9.91284051e-01,
       9.91284051e-01, 9.91284051e-01, 9.95331797e-01, 9.95331797e-01,
       9.95361343e-01, 9.97074987e-01, 9.97074987e-01, 9.97074987e-01,
       9.97193169e-01, 9.97222715e-01, 9.97281806e-01, 9.97340897e-01,
       9.97340897e-01, 9.97399988e-01, 9.97399988e-01, 9.97399988e-01,
       9.97399988e-01, 9.98109082e-01, 9.99438634e-01, 9.99438634e-01,
       9.99527271e-01, 9.99527271e-01, 9.99556816e-01, 9.99586362e-01,
       9.99586362e-01, 9.99615907e-01, 9.99615907e-01, 9.99615907e-01,
       9.99645453e-01, 9.99645453e-01, 9.99645453e-01, 1.00000000e+00])
```

In [33]:

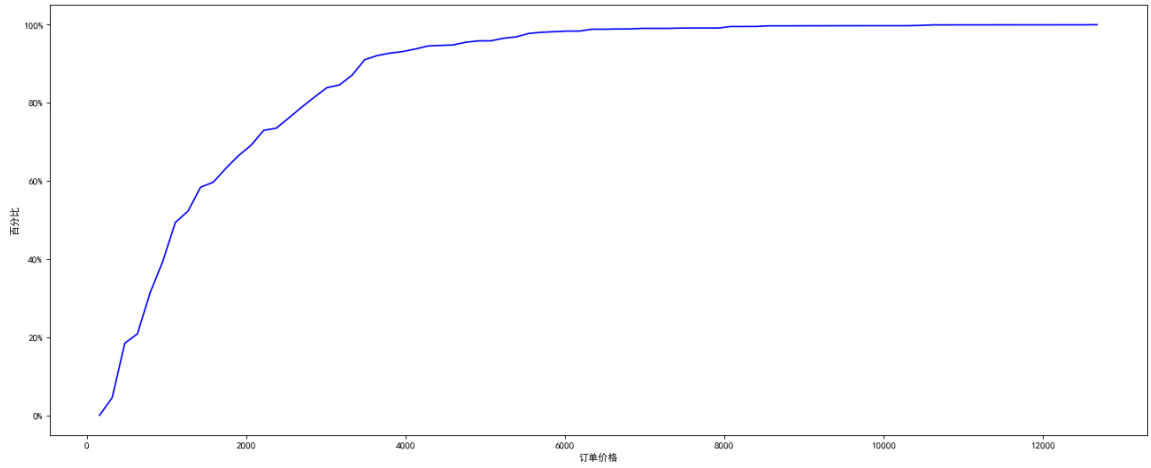
```
bin_edges_plot = np.delete(bin_edges, 0)
```

In [34]:

```
plt.figure(figsize=(20,8), dpi=80)
plt.xlabel('订单价格')
plt.ylabel('百分比')

plt.style.use('ggplot')

def to_percent(temp, position):
    return '%1.0f'%(100*temp) + '%'
plt.gca().yaxis.set_major_formatter(FuncFormatter(to_percent))

plt.plot(bin_edges_plot, hist_per, color='blue')
```

Out[34]:

```
[<matplotlib.lines.Line2D at 0x203bdbddac8>]
```

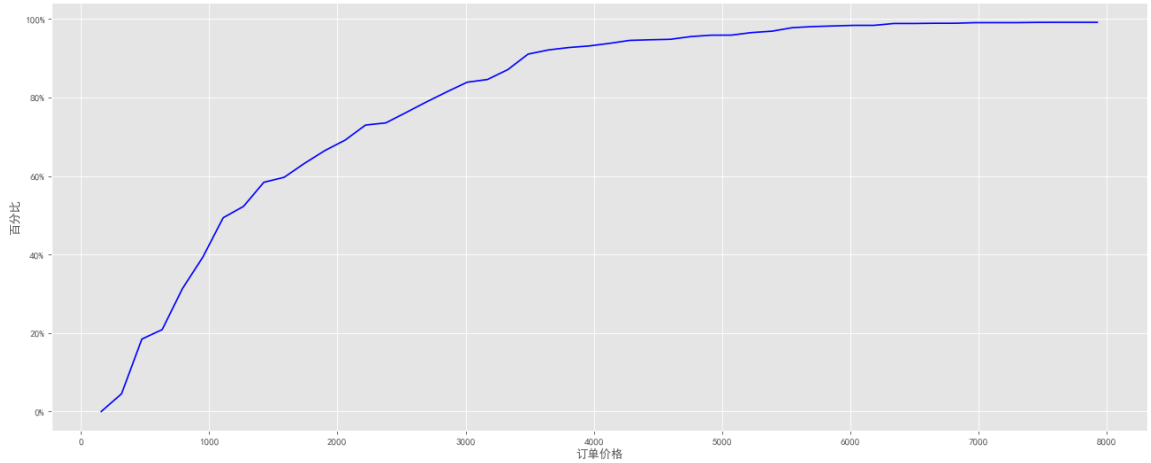


In [35]:

```
plt.figure(figsize=(20,8), dpi=80)
plt.xlabel('订单价格')
plt.ylabel('百分比')

plt.style.use('ggplot')

def to_percent(temp, position):
    return '%1.0f'%(100*temp) + '%'
plt.gca().yaxis.set_major_formatter(FuncFormatter(to_percent))

plt.plot(bin_edges_plot[:50], hist_per[:50], color='blue')
```

Out[35]:

```
[<matplotlib.lines.Line2D at 0x203be4a5438>]
```



微观分析

有效订单量

In [36]:

```
df3 = df2.copy()
df3['order_time_hms'] = df3['sale_ord_tm'].apply(lambda x: x.strftime('%H:00:00'))
```

In [37]:

```
df3
```

Out[37]:

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
0	linfbi007	116828823929	116828823929	2020-05-25 18:09:39	2020-05-25	1000(
1	13601089905_p	116769479986	121562216719	2020-05-25 00:04:15	2020-05-25	1000(
2	jd_UbSjKwFGOfbv	116815391384	116809219025	2020-05-25 13:47:33	2020-05-25	1000(
3	yangwangjun1300	116814673181	116814673181	2020-05-25 14:34:25	2020-05-25	1000(
4	jd_77dbadc203044	116811074034	116811074034	2020-05-25 14:47:42	2020-05-25	1000(
5	葑存记忆	121591740399	121591740399	2020-05-25 11:38:12	2020-05-25	1000(
6	jd_uuwhuTZixluQ	116825666739	116825666739	2020-05-25 17:09:30	2020-05-25	1000(
7	haiqiang0307	116881662586	116881662586	2020-05-25 23:24:22	2020-05-25	1000(

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
8	shangchunping0216	116847011379	116847011379	2020-05-25 20:12:12	2020-05-25	1000
9	jd_5d3fef6659091	116835354680	116835354680	2020-05-25 17:20:53	2020-05-25	1000
10	9657401-591066	116810188022	116810188022	2020-05-25 14:17:39	2020-05-25	1000
12	maxiaohong1985	116785989012	116785989012	2020-05-25 08:22:11	2020-05-25	1000
16	jd_76b772039a54e	121623032619	121623032619	2020-05-25 00:01:56	2020-05-25	1000
18	jd_67eda241885a8	116775076123	116775076123	2020-05-25 10:03:11	2020-05-25	1000
19	mxd_816	116795056082	116795056082	2020-05-25 11:26:17	2020-05-25	1000
21	wlx1993	116803837055	116803837055	2020-05-25 12:38:17	2020-05-25	1000

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
23	TracyJordan123	116791411932	121647436968	2020-05-25 10:15:06	2020-05-25	1000(
24	jd_6569c2782e2e9	116830314520	116830314968	2020-05-25 16:33:37	2020-05-25	1000(
25	suihjbuy	116866820625	116866820625	2020-05-25 22:13:09	2020-05-25	1000(
27	287122990_m	116810099005	116810099005	2020-05-25 13:38:34	2020-05-25	1000(
28	jd_4ab09a8c03310	116878273919	121593736079	2020-05-25 23:29:19	2020-05-25	1000(
29	924288726_m	116827329014	116829399068	2020-05-25 17:18:08	2020-05-25	1000(
30	jd_5e55d9ba72e6e	121586518318	121586518318	2020-05-25 10:32:37	2020-05-25	1000(
31	13509924666_p	116806504241	116806504241	2020-05-25 13:36:09	2020-05-25	1000(

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
33	jd_46710dcc309de	121582920653	121582920653	2020-05-25 09:12:58	2020-05-25	1000
35	jd_6818c0f2359c0	121587377231	121587377231	2020-05-25 10:42:26	2020-05-25	1000
39	linda_wang1983	116867809616	116867809616	2020-05-25 22:22:44	2020-05-25	1000
40	646946256_m	116815451135	116815451135	2020-05-25 14:55:39	2020-05-25	1000
42	510902ZMD011	121626001256	121626001256	2020-05-25 00:20:05	2020-05-25	1000
45	18523242420_p	121621504288	121621504288	2020-05-25 00:00:41	2020-05-25	1000
...	...	...	...	...	...	
76580	15858928700_p	116858634164	116857496566	2020-05-25 21:13:10	2020-05-25	
76581	jd_7828e3d70bd3c	116824724663	116824724663	2020-05-25 16:47:00	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76582	jd_71wCRu5PJZvc	116849971285	116835931163	2020-05-25 20:21:46	2020-05-25	
76584	jd_yvxaSRfpnSDQ	116829815261	116813452667	2020-05-25 17:13:10	2020-05-25	
76586	jd_60208b43fe275	116820137009	116820137009	2020-05-25 16:10:48	2020-05-25	
76587	阿苗耐的住寂寞	116784770491	116784770491	2020-05-25 11:53:24	2020-05-25	
76590	18933954534_p	116837089872	116845427544	2020-05-25 18:49:49	2020-05-25	
76592	yuexiafeiyingde	121639738249	121639738249	2020-05-25 07:49:57	2020-05-25	
76593	511023ZMD888	121621148161	121621148161	2020-05-25 00:01:33	2020-05-25	
76596	jd_fZvCgSxZhXcQ	116861787953	116869875256	2020-05-25 21:41:45	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76597	jd_63ac4fbb06ebd	116865963256	116865963256	2020-05-25 21:18:40	2020-05-25	
76598	jd_wggqMlNwcKZu	116828835568	116828729619	2020-05-25 17:37:01	2020-05-25	
76599	jd_45370c1b9c0fc	116805964089	116805964089	2020-05-25 14:18:12	2020-05-25	
76601	jd_696d1396be5b1	116798619729	116798619729	2020-05-25 12:05:12	2020-05-25	
76602	红樱桃JD	116864510111	116864510111	2020-05-25 21:43:33	2020-05-25	
76604	jd_SIuXCfEaqtcC	116849725400	116843295444	2020-05-25 19:31:07	2020-05-25	
76605	18608085791_p	121626875846	121626875846	2020-05-25 00:15:05	2020-05-25	
76608	215414ZMD001	121562570798	121562570798	2020-05-25 00:04:22	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76609	aleeyoung	116825271571	116825271571	2020-05-25 17:05:12	2020-05-25	
76611	jd_69424511b4ac2	116817061872	116817061872	2020-05-25 15:43:53	2020-05-25	
76612	513028ZMD000	121563801487	121563801487	2020-05-25 00:25:04	2020-05-25	
76613	jd_688188e631142	116833414224	116833414224	2020-05-25 18:17:48	2020-05-25	
76614	350127ZMD999	116814845655	116814845655	2020-05-25 15:03:17	2020-05-25	
76617	jd_6091df073a467	116868694552	116868694552	2020-05-25 21:34:48	2020-05-25	
76618	jd_4209b7a5111f3	116878457077	116878457077	2020-05-25 23:46:19	2020-05-25	
76621	513401ZMD001	116786238803	116786238803	2020-05-25 09:33:51	2020-05-25	

	user_log_acct	parent_sale_ord_id	sale_ord_id	sale_ord_tm	sale_ord_dt	iter
76624	jd_7b852d8fa7721	116798024469	116798024469	2020-05-25 11:40:35	2020-05-25	
76625	350583ZMD066	116804271579	116804271579	2020-05-25 15:40:45	2020-05-25	
76626	sd32513500	116849959579	116849959579	2020-05-25 21:48:53	2020-05-25	
76629	441900ZMD666	116841557298	116841557298	2020-05-25 19:40:55	2020-05-25	

33846 rows × 26 columns

In [38]:

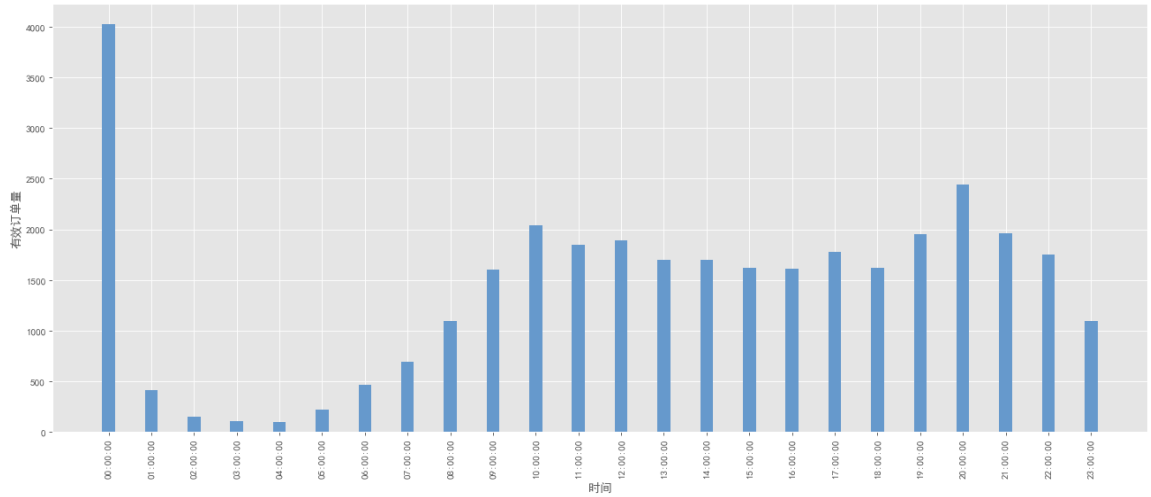
```
pay_time_df = df3.groupby('order_time_hms')['sale_ord_id'].count()
pay_time_df
```

Out[38]:

```
order_time_hms
00:00:00    4032
01:00:00     406
02:00:00     146
03:00:00     101
04:00:00      93
05:00:00     216
06:00:00     461
07:00:00     692
08:00:00    1091
09:00:00    1602
10:00:00    2043
11:00:00    1850
12:00:00    1893
13:00:00    1696
14:00:00    1697
15:00:00    1620
16:00:00    1607
17:00:00    1774
18:00:00    1620
19:00:00    1953
20:00:00    2445
21:00:00    1957
22:00:00    1753
23:00:00    1098
Name: sale_ord_id, dtype: int64
```

In [39]:

```
x = pay_time_df.index
y = pay_time_df.values

plt.figure(figsize=(20,8),dpi=80)
plt.style.use('ggplot')
plt.xlabel('时间')
plt.ylabel("有效订单量")
plt.xticks(range(len(x)), x, rotation=90)
rect = plt.bar(x, y, width=0.3, color=['#6699CC'])
```



时间维度来对订单数据进行拆分-人均有效订单量

In [40]:

```
order_time_df = df3.groupby('order_time_hms')['sale_ord_id'].agg({'order_num':'count'})
order_time_df
```

Out[40]:

order_time_hms	order_num
00:00:00	4032
01:00:00	406
02:00:00	146
03:00:00	101
04:00:00	93
05:00:00	216
06:00:00	461
07:00:00	692
08:00:00	1091
09:00:00	1602
10:00:00	2043
11:00:00	1850
12:00:00	1893
13:00:00	1696
14:00:00	1697
15:00:00	1620
16:00:00	1607
17:00:00	1774
18:00:00	1620
19:00:00	1953
20:00:00	2445
21:00:00	1957
22:00:00	1753
23:00:00	1098

In [41]:

```
user_time_df = df3.groupby('order_time_hms')['user_log_acct'].agg({'user_num':'nunique'})
user_time_df
```

Out[41]:

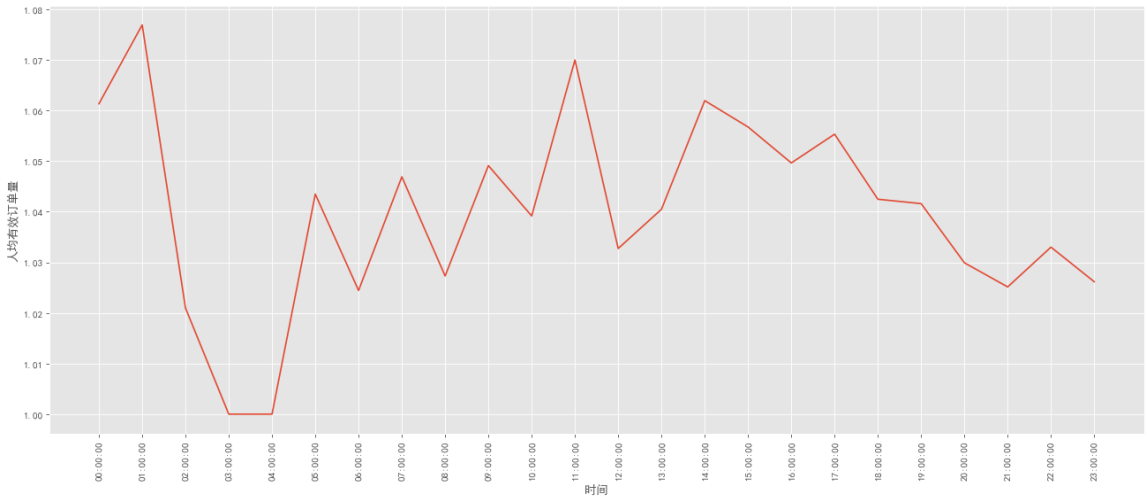
order_time_hms	user_num
00:00:00	3799
01:00:00	377
02:00:00	143
03:00:00	101
04:00:00	93
05:00:00	207
06:00:00	450
07:00:00	661
08:00:00	1062
09:00:00	1527
10:00:00	1966
11:00:00	1729
12:00:00	1833
13:00:00	1630
14:00:00	1598
15:00:00	1533
16:00:00	1531
17:00:00	1681
18:00:00	1554
19:00:00	1875
20:00:00	2374
21:00:00	1909
22:00:00	1697
23:00:00	1070

In [42]:

```
order_num_per_user = order_time_df['order_num'] / user_time_df['user_num']

x = order_num_per_user.index
y = order_num_per_user.values

plt.figure(figsize=(20,8),dpi=80)
plt.style.use('ggplot')
plt.xlabel('时间')
plt.ylabel("人均有效订单量")
plt.xticks(range(len(x)),x,rotation=90)
plt.plot(x, y)
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x203bdffc588>]
```



客单价和平均订单价格

In [43]:

```
total_pay_time_df = df3.groupby('order_time_hms')['total_actual_pay'].agg({'total_pay':'sum'})
total_pay_time_df
```

Out[43]:

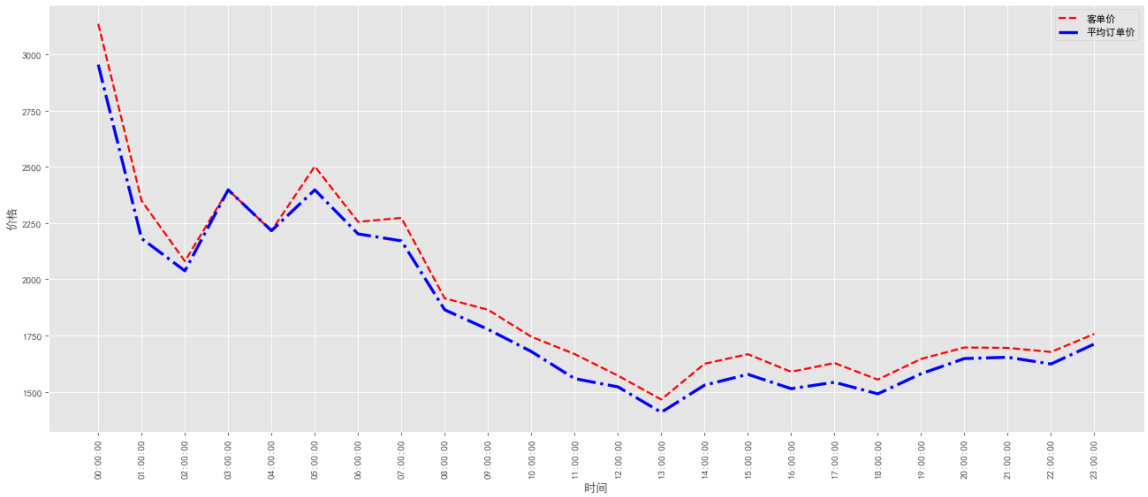
order_time_hms	total_pay
00:00:00	11909925.00
01:00:00	885705.70
02:00:00	297508.92
03:00:00	242141.00
04:00:00	206095.00
05:00:00	517820.23
06:00:00	1014900.00
07:00:00	1502330.99
08:00:00	2034257.26
09:00:00	2847787.61
10:00:00	3430583.32
11:00:00	2883861.80
12:00:00	2881552.11
13:00:00	2389907.35
14:00:00	2596434.98
15:00:00	2556149.88
16:00:00	2433215.77
17:00:00	2736634.54
18:00:00	2415622.73
19:00:00	3086872.52
20:00:00	4029216.39
21:00:00	3236355.95
22:00:00	2846488.06
23:00:00	1880491.04

In [44]:

```
pay_per_user = total_pay_time_df['total_pay'] / user_time_df['user_num'] # 客单价：销售额 / 顾客数
pay_per_order = total_pay_time_df['total_pay'] / order_time_df['order_num'] # 平均订单价：销售额 / 订单数

x = pay_per_user.index
y = pay_per_user.values
y2 = pay_per_order.values

plt.figure(figsize=(20,8),dpi=80)
plt.style.use('ggplot')
plt.xlabel('时间')
plt.ylabel("价格")
plt.xticks(range(len(x)),x,rotation=90)

plt.plot(x, y, color='red',linewidth=2.0,linestyle='--')
plt.plot(x, y2, color='blue',linewidth=3.0,linestyle='-.')
plt.legend(['客单价','平均订单价'])
```

Out[44]:

<matplotlib.legend.Legend at 0x203be09ec88>



价格累积分布图

In [45]:

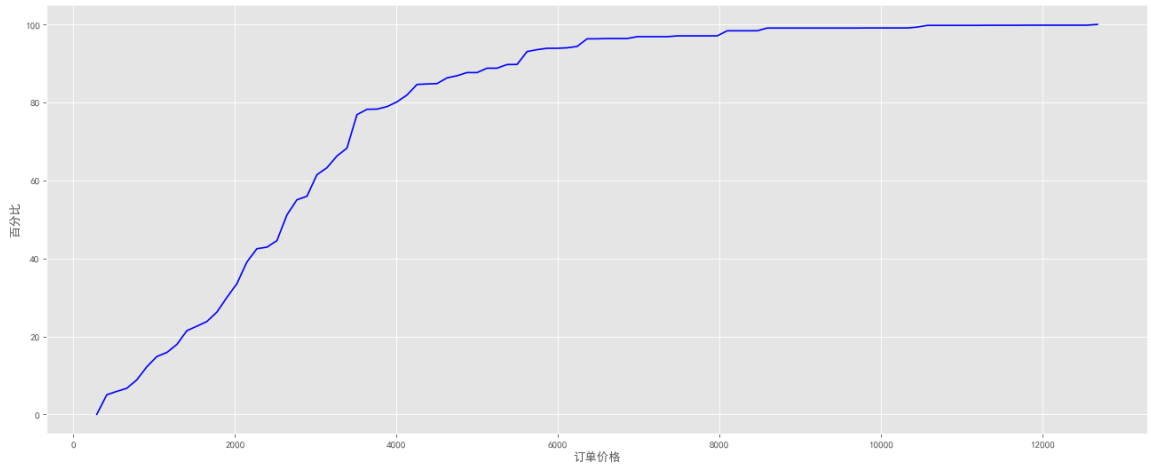
```
df4 = df3.copy()
df5 = df3.copy()

df4 = df4[df4['order_time_hms'] == '00:00:00']
df5 = df5[df5['order_time_hms'] == '20:00:00']
```

In [46]:

```
def plot_acc_line(price_series, bin_num):
    len = price_series.count()
    hist, bin_edges = np.histogram(price_series, bins=bin_num) #生成直方图函数
    hist_sum = np.cumsum(hist)
    hist_per = hist_sum / len * 100
    hist_per_plot = np.insert(hist_per, 0, 0)

    plt.figure(figsize=(20,8), dpi=80)
    plt.xlabel('订单价格')
    plt.ylabel('百分比')

    plt.plot(bin_edges, hist_per_plot, color='blue')
```

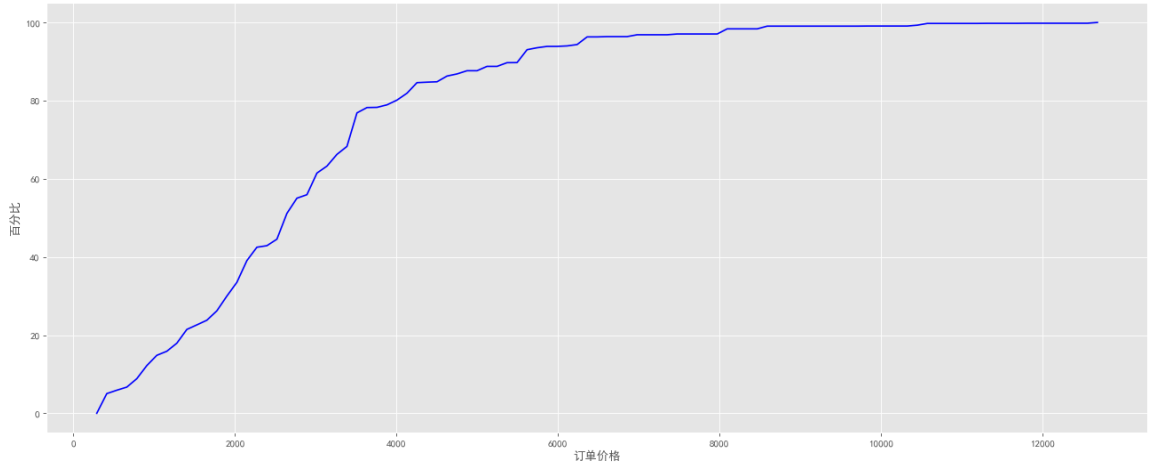
In [47]:

```
# 0时价格累积分布折线图

price_series_0 = df4['after_prefr_unit_price']
plot_acc_line(price_series_0, 100)
```



In [48]:

```
# 20时价格累积分布折线图

price_series_20 = df5['after_prefr_unit_price']
plot_acc_line(price_series_0, 100)
```



从时间维度对订单进行拆分

In [49]:

```
#0时的优惠订单数
offer_order_0 = df4['sale_ord_id'][df4['total_offer_amount'] > 0].count()

#0时订单数
order_num_0 = df4['sale_ord_id'].count()

#0时优惠订单比
offer_order_per_0 = offer_order_0 / order_num_0

print('0时的优惠订单数:{}, 0时的订单数:{}, 优惠订单比例：{}'.format(offer_order_0, order_num_0, offer_order_per_0))
```

0时的优惠订单数:3788, 0时的订单数:4032, 优惠订单比例：0.939484126984127

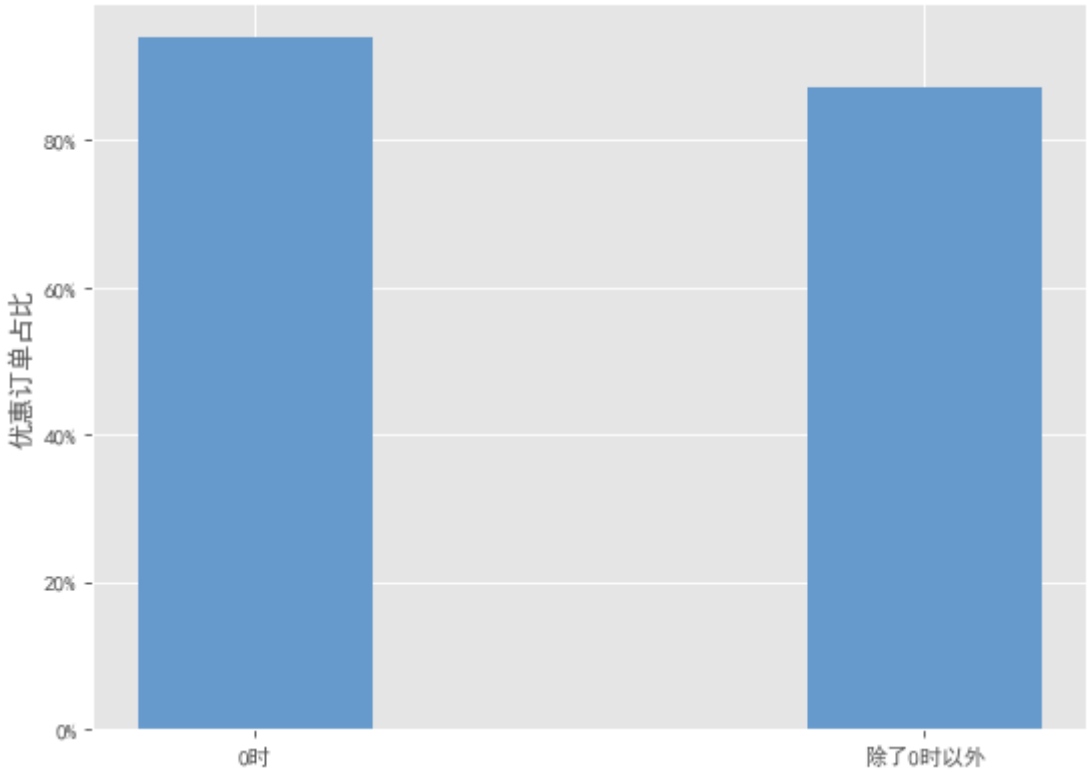
In [50]:

```
#全部优惠订单数
offer_order_all = df3['sale_ord_id'][df3['total_offer_amount'] > 0].count()

#全部订单数
order_all = df3['sale_ord_id'].count()

#其他时间优惠订单数
offer_order_other = offer_order_all - offer_order_0

#其他时间订单数
order_num_other = order_all - order_num_0

offer_order_per_other = offer_order_other / order_num_other

print('其他时间的优惠订单数:{}, 其他时间的订单数:{}, 其他时间优惠订单比例：{}'.format(offer_order_other, order_num_other, offer_order_per_other))
```

其他时间的优惠订单数:25983, 其他时间的订单数:29814, 其他时间优惠订单比例：0.871503
3205876433

In [51]:

```
#0时和其他时间的优惠订单的占比对比:可视化

plt.figure(figsize=(8, 6), dpi=80)
N = 2
index = ('0时', '除了0时以外')
data = (offer_order_per_0, offer_order_per_other)
width = 0.35
plt.ylabel("优惠订单占比")

def to_percent(temp, position):
    return '%1.0f'%(100*temp) + '%'
plt.gca().yaxis.set_major_formatter(FuncFormatter(to_percent))

p2 = plt.bar(index, data, width, color='#6699CC')
```



In [52]:

```
total_pay_time_df = df3.groupby('order_time_hms')['total_offer_amount'].agg({'total_offer_amount':'sum'})
total_pay_time_df
```

Out[52]:

order_time_hms	total_offer_amount
00:00:00	2773061.00
01:00:00	245842.00
02:00:00	64832.00
03:00:00	49619.00
04:00:00	42450.00
05:00:00	109581.00
06:00:00	215421.00
07:00:00	347409.00
08:00:00	555511.00
09:00:00	678265.00
10:00:00	805570.97
11:00:00	692779.99
12:00:00	725679.99
13:00:00	608480.98
14:00:00	622332.99
15:00:00	581270.98
16:00:00	617612.00
17:00:00	624794.97
18:00:00	632697.98
19:00:00	752224.98
20:00:00	935777.99
21:00:00	779780.96
22:00:00	705570.99
23:00:00	479446.98

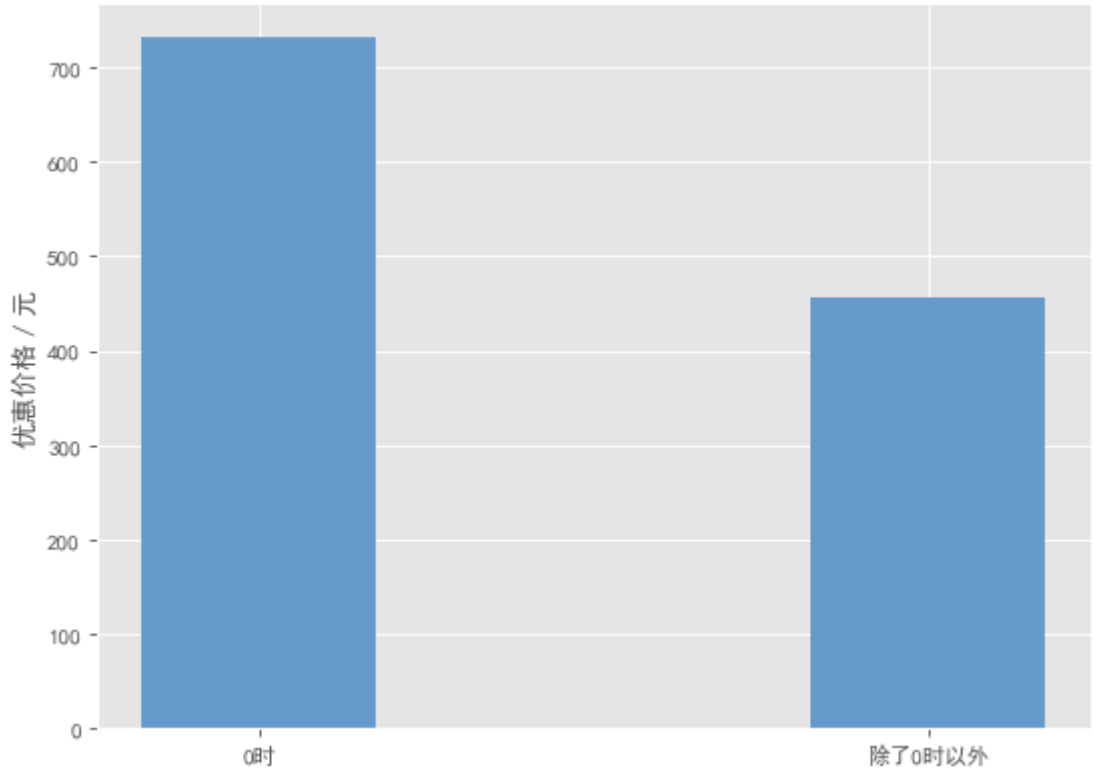
In [53]:

```
offer_amount_0 = total_pay_time_df['total_offer_amount'][0]

offer_amount_other = total_pay_time_df[1:].apply(lambda x: x.sum())['total_offer_amount'] #按行求和

offer_amount_0_avg = offer_amount_0 / offer_order_0
offer_amount_other_avg = offer_amount_other / offer_order_other

print('0时平均优惠价格:{}, 其他时间平均优惠价格:{}'.format(offer_amount_0_avg, offer_amount_other_avg))
```

0时平均优惠价格:732.0646779303062, 其他时间平均优惠价格:456.9508043720895

In [54]:

```
#0时和其他时间的平均优惠价格对比：可视化
plt.figure(figsize=(8, 6), dpi=80)
N = 2
index = ('0时', '除了0时以外')

values = (offer_amount_0_avg, offer_amount_other_avg)
width = 0.35

plt.ylabel("优惠价格 / 元")

p2 = plt.bar(index, values, width, color='#6699CC')
```



从地区维度对订单进行拆分

In [55]:

```
df6 = df2.copy()

order_area_df = df6.groupby('user_site_province_id', as_index=False)['sale_ord_id'].agg({'order_num':'count'})
order_area_df.columns = ['province_id','order_num']
order_area_df
```

Out[55]:

	province_id	order_num
0	0	9883
1	1	1180
2	10	88
3	11	40
4	12	714
5	13	394
6	14	194
7	15	444
8	16	254
9	17	331
10	18	181
11	19	1831
12	2	592
13	20	130
14	21	112
15	22	557
16	23	51
17	24	62
18	25	94
19	26	4
20	27	289
21	28	41
22	29	7
23	3	142
24	30	9
25	31	49
26	32	1
27	4	101
28	42	2
29	5	274
30	6	111
31	7	294
32	8	265
33	9	82
34	Not Given	15043

In [56]:

```
order_area_df.drop([34], inplace=True)
order_area_df['province_id']=order_area_df['province_id'].astype('int')
order_area_df
```

Out[56]:

	province_id	order_num
0	0	9883
1	1	1180
2	10	88
3	11	40
4	12	714
5	13	394
6	14	194
7	15	444
8	16	254
9	17	331
10	18	181
11	19	1831
12	2	592
13	20	130
14	21	112
15	22	557
16	23	51
17	24	62
18	25	94
19	26	4
20	27	289
21	28	41
22	29	7
23	3	142
24	30	9
25	31	49
26	32	1
27	4	101
28	42	2
29	5	274
30	6	111
31	7	294
32	8	265
33	9	82

In [57]:

```
city = 'city_level.csv'
df_city = pd.read_csv(city,sep = ',', encoding="gbk", dtype=str)
df_city['province_id'] = df_city['province_id'].astype('int')
df_city
```

Out[57]:

	dim_city_id	dim_city_name	city_id	dim_province_id	dim_province_name	province_
0	1000	济南市	1000	13	山东	
1	1007	青岛市	1007	13	山东	
2	1016	淄博市	1016	13	山东	
3	1022	枣庄市	1022	13	山东	
4	1025	东营市	1025	13	山东	
5	1032	潍坊市	1032	13	山东	
6	1042	烟台市	1042	13	山东	
7	1053	威海市	1053	13	山东	
8	1058	莱芜市	1058	13	山东	
9	1060	德州市	1060	13	山东	
10	1072	临沂市	1072	13	山东	
11	1081	聊城市	1081	13	山东	
12	1090	滨州市	1090	13	山东	
13	1099	菏泽市	1099	13	山东	
14	1108	日照市	1108	13	山东	
15	1112	泰安市	1112	13	山东	
16	1114	铜陵市	1114	14	安徽	
17	1116	合肥市	1116	14	安徽	
18	1121	淮南市	1121	14	安徽	
19	1124	淮北市	1124	14	安徽	
20	1127	芜湖市	1127	14	安徽	
21	113	万州区	113	4	重庆	
22	1132	蚌埠市	1132	14	安徽	
23	1137	马鞍山市	1137	14	安徽	
24	114	涪陵区	114	4	重庆	
25	1140	安庆市	1140	14	安徽	
26	115	梁平区	115	4	重庆	
27	1151	黄山市	1151	14	安徽	
28	1158	宁波市	1158	15	浙江	
29	1159	滁州市	1159	14	安徽	
...	...	...	...	...	...	
448	776	黑河市	776	10	黑龙江	
449	78	黄浦区	78	2	上海	
450	78	黄浦区	78	2	上海	

	dim_city_id	dim_city_name	city_id	dim_province_id	dim_province_name	province_
451	782	绥化市	782	10	黑龙江	
452	793	大兴安岭地区	793	10	黑龙江	
453	799	呼和浩特市	799	11	内蒙古	
454	805	包头市	805	11	内蒙古	
455	810	乌海市	810	11	内蒙古	
456	812	赤峰市	812	11	内蒙古	
457	823	乌兰察布市	823	11	内蒙古	
458	835	锡林郭勒盟	835	11	内蒙古	
459	848	呼伦贝尔市	848	11	内蒙古	
460	870	鄂尔多斯市	870	11	内蒙古	
461	880	巴彦淖尔市	880	11	内蒙古	
462	891	阿拉善盟	891	11	内蒙古	
463	895	兴安盟	895	11	内蒙古	
464	902	通辽市	902	11	内蒙古	
465	904	南京市	904	12	江苏	
466	911	徐州市	911	12	江苏	
467	919	连云港市	919	12	江苏	
468	925	淮安市	925	12	江苏	
469	933	宿迁市	933	12	江苏	
470	939	盐城市	939	12	江苏	
471	951	扬州市	951	12	江苏	
472	959	泰州市	959	12	江苏	
473	965	南通市	965	12	江苏	
474	972	镇江市	972	12	江苏	
475	978	常州市	978	12	江苏	
476	984	无锡市	984	12	江苏	
477	988	苏州市	988	12	江苏	

478 rows × 7 columns

In [58]:

```
df_city = df_city.drop_duplicates(subset=['province_id'], keep='first') # 保留重复数据的第一个，也就是只保留省份数据
df_city
```

Out[58]:

	dim_city_id	dim_city_name	city_id	dim_province_id	dim_province_name	province_id
0	1000	济南市	1000	13	山东	13
16	1114	铜陵市	1114	14	安徽	14
21	113	万州区	113	4	重庆	4
28	1158	宁波市	1158	15	浙江	15
51	1303	福州市	1303	16	福建	16
53	1310	钓鱼岛	1310	84	钓鱼岛	84
69	1381	武汉市	1381	17	湖北	17
77	142	石家庄市	142	5	河北	5
87	1482	长沙市	1482	18	湖南	18
101	15945	阿拉尔市	15945	31	新疆	31
103	1601	广州市	1601	19	广东	19
125	1715	南宁市	1715	20	广西	20
137	1827	南昌市	1827	21	江西	21
148	1930	成都市	1930	22	四川	22
170	2121	海口市	2121	23	海南	23
171	2144	贵阳市	2144	24	贵州	24
180	2235	昆明市	2235	25	云南	25
196	2376	西安市	2376	27	陕西	27
208	2487	兰州市	2487	28	甘肃	28
222	2580	西宁市	2580	29	青海	29
230	2628	银川市	2628	30	宁夏	30
252	2768	台湾	2768	32	台湾	32
253	2780	济源市	2780	7	河南	7
254	2800	海淀区	2800	1	北京	1
266	2813	徐汇区	2813	2	上海	2
301	2951	拉萨市	2951	26	西藏	26
306	2992	辽源市	2992	9	吉林	9
307	303	太原市	303	6	山西	6
388	51035	东丽区	51035	3	天津	3
409	52994	香港特别行政区	52994	52993	港澳	52993
415	560	沈阳市	560	8	辽宁	8
437	698	哈尔滨市	698	10	黑龙江	10
453	799	呼和浩特市	799	11	内蒙古	11
465	904	南京市	904	12	江苏	12

In [59]:

```
df_city = df_city[['province_id','dim_province_name']].sort_values(by='province_id',ascending=True).reset_index()
df_city.drop(['index'], axis=1, inplace=True)
df_city
```

Out[59]:

	province_id	dim_province_name
0	1	北京
1	2	上海
2	3	天津
3	4	重庆
4	5	河北
5	6	山西
6	7	河南
7	8	辽宁
8	9	吉林
9	10	黑龙江
10	11	内蒙古
11	12	江苏
12	13	山东
13	14	安徽
14	15	浙江
15	16	福建
16	17	湖北
17	18	湖南
18	19	广东
19	20	广西
20	21	江西
21	22	四川
22	23	海南
23	24	贵州
24	25	云南
25	26	西藏
26	27	陕西
27	28	甘肃
28	29	青海
29	30	宁夏
30	31	新疆
31	32	台湾
32	84	钓鱼岛
33	52993	港澳

In [60]:

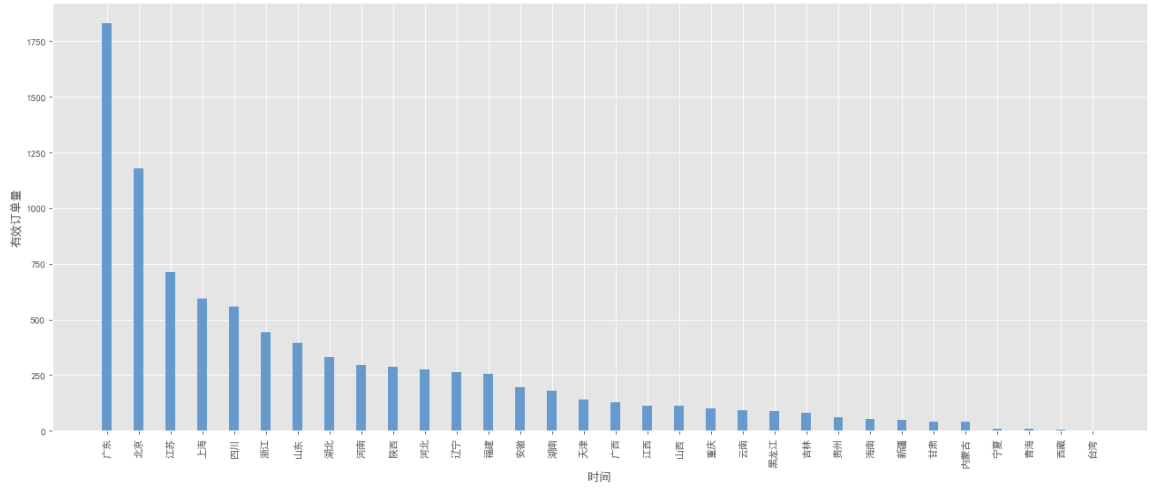
```
order_province_df = pd.merge(order_area_df, df_city, on='province_id').sort_values(by='order_num', ascending=False)
order_province_df
```

Out[60]:

	province_id	order_num	dim_province_name
10	19	1831	广东
0	1	1180	北京
3	12	714	江苏
11	2	592	上海
14	22	557	四川
6	15	444	浙江
4	13	394	山东
8	17	331	湖北
29	7	294	河南
19	27	289	陕西
27	5	274	河北
30	8	265	辽宁
7	16	254	福建
5	14	194	安徽
9	18	181	湖南
22	3	142	天津
12	20	130	广西
13	21	112	江西
28	6	111	山西
26	4	101	重庆
17	25	94	云南
1	10	88	黑龙江
31	9	82	吉林
16	24	62	贵州
15	23	51	海南
24	31	49	新疆
20	28	41	甘肃
2	11	40	内蒙古
23	30	9	宁夏
21	29	7	青海
18	26	4	西藏
25	32	1	台湾

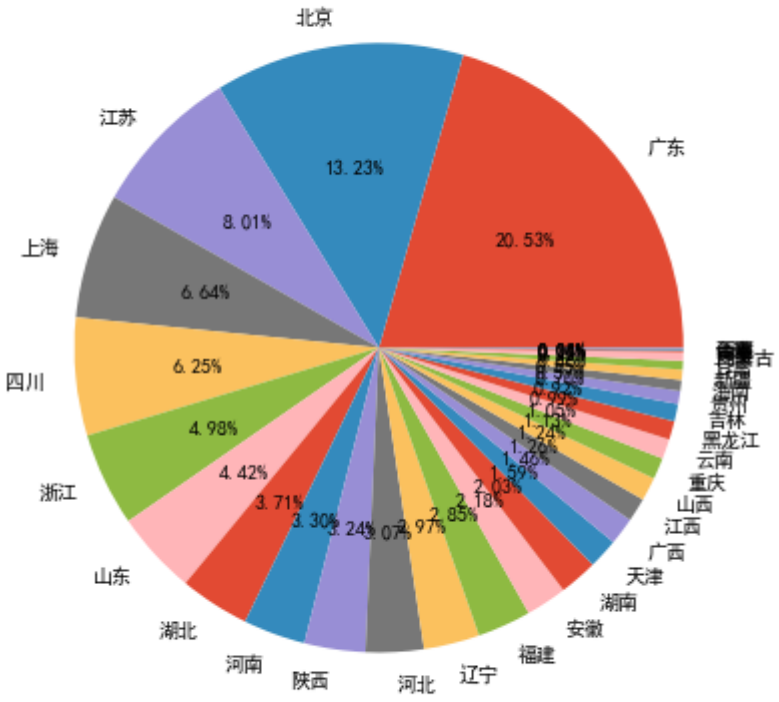
In [61]:

```
#有效订单量
plt.style.use('ggplot')

x = order_province_df['dim_province_name']
y = order_province_df['order_num']

plt.figure(figsize=(20,8),dpi=80)
plt.style.use('ggplot')
plt.xlabel('时间')
plt.ylabel("有效订单量")
plt.xticks(range(len(x)), x, rotation=90)
rect = plt.bar(x, y, width=0.3, color=['#6699CC'])
```



In [62]:

```
#有效订单量-饼图
plt.figure(figsize=(6,9))
labels = order_province_df['dim_province_name']

plt.pie(order_province_df['order_num'], labels=labels,autopct='%1.2f%%') # autopct :控制饼图内百分比设置, '%1.1f'指小数点前后位数(没有用空格补齐);

plt.axis('equal')
plt.show()
```



In [63]:

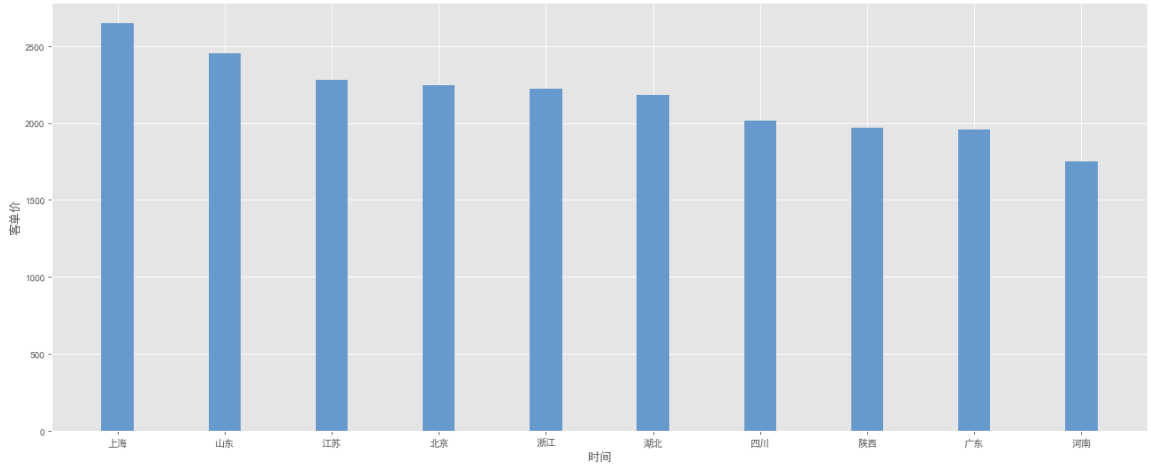
```
#各省份客单价对比

cust_price_df = df6.groupby('user_site_province_id', as_index=False)['total_actual_pay'].agg({'total_pay':'sum'})
cust_price_df.columns = ['province_id','total_pay']
cust_price_df.drop([34], inplace=True)
cust_price_df['province_id'] = cust_price_df['province_id'].astype('int')
cust_price_df = pd.merge(cust_price_df, df_city, on='province_id').sort_values(by='total_pay', ascending=False)
cust_price_df['order_num'] = order_province_df['order_num']

cust_df = df6.groupby('user_site_province_id', as_index=False)['user_log_acct'].agg({'user_num':'nunique'})
cust_df.columns = ['province_id','user_num']
cust_df.drop([34], inplace=True)
cust_df['province_id'] = cust_df['province_id'].astype('int')

cust_price_df = pd.merge(cust_price_df, cust_df, on='province_id')
cust_price_df['cust_price'] = cust_price_df['total_pay'] / cust_price_df['user_num'] #计算客单价
cust_price_df = cust_price_df.sort_values(by='order_num', ascending=False)
cust_price_df = cust_price_df[:10]
cust_price_df = cust_price_df.sort_values(by='cust_price', ascending=False)

cust_price_df
```

Out[63]:

	province_id	total_pay	dim_province_name	order_num	user_num	cust_price
3	2	1425442.00	上海	592	538	2649.520446
5	13	938661.01	山东	394	383	2450.812037
2	12	1603704.00	江苏	714	704	2277.988636
1	1	2548956.74	北京	1180	1135	2245.776863
6	15	937725.00	浙江	444	422	2222.097156
7	17	712541.00	湖北	331	327	2179.024465
4	22	1104843.00	四川	557	548	2016.136861
9	27	564524.00	陕西	289	287	1966.982578
0	19	3547611.01	广东	1831	1813	1956.762830
11	7	499252.64	河南	294	285	1751.763649

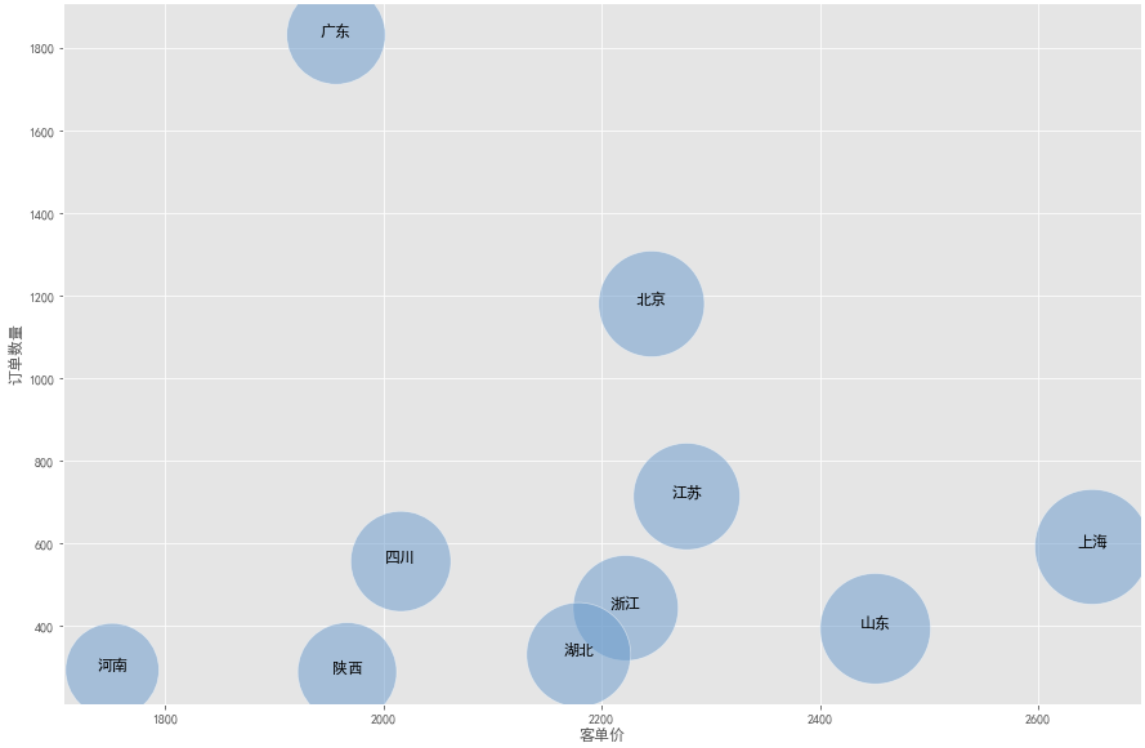
In [64]:

```
plt.style.use('ggplot')

x = cust_price_df['dim_province_name']
y = cust_price_df['cust_price']

plt.figure(figsize=(20,8),dpi=80)
plt.style.use('ggplot')
plt.xlabel('时间')
plt.ylabel("客单价")
rect = plt.bar(x, y, width=0.3, color=['#6699CC'])
```



In [65]:

```
plt.figure(figsize = (15,10))

x = cust_price_df['cust_price']
y = cust_price_df['order_num']

ax = sns.scatterplot(x, y, alpha=0.5, s=x*3, c=['#6699CC']) # 绘制气泡图. alpha是不透明度
ax.set_xlabel("客单价",fontsize=12)
ax.set_ylabel("订单数量",fontsize=12)

province_list = [3,5,2,1,6,7,4,9,0,11]
# 在气泡上加文字
for line in province_list:
    ax.text(x[line], y[line], cust_price_df['dim_province_name'][line], horizontalalignment='center', size='large', color='black', weight='semibold')
```



头部省份的四个品牌的渗透率

In [66]:

```
#不同品牌的产品单价
df7 = df2.copy()

brand_sale_df = df7.groupby('brandname', as_index=False).agg({'total_actual_pay':'sum', 'sale_qtty':'sum'}).sort_values(by='total_actual_pay', ascending=False)
brand_sale_df
```

Out[66]:

	brandname	total_actual_pay	sale_qtty
30	海尔（Haier）	16634130.64	7540
13	容声（Ronshen）	11813300.63	5989
37	西门子（SIEMENS）	6738061.09	1260
35	美的（Midea）	4473746.14	2709
29	海信（Hisense）	2663095.57	1720
17	康佳（KONKA）	2276290.54	2584
25	松下（Panasonic）	2146912.00	453
36	美菱（MeiLing）	1787125.73	890
0	TCL	1578416.85	1419
27	格力（GREE）	1426076.00	726
7	创维（Skyworth）	1123975.88	1023
8	华凌	1008649.00	848
18	志高（CHIGO）	946167.00	2267
10	卡萨帝（Casarte）	871072.00	88
22	新飞（Frestec）	789853.84	1011
11	奥克斯（AUX）	680361.00	1443
9	博世（BOSCH）	604907.00	93
4	云米（VIOMI）	571193.92	241
34	统帅（Leader）	567603.55	472
12	奥马（Homa）	365279.59	270
14	小天鹅（LittleSwan）	361282.18	292
33	现代（HYUNDAI）	327445.00	813
24	日普（RIPU）	151780.00	470
5	伊莱克斯（Electrolux）	110349.00	51
2	三星（SAMSUNG）	105273.00	27
32	熊猫（PANDA）	97140.00	315
6	冰熊（bingxiong）	96340.00	305
31	澳柯玛（AUCMA）	79482.00	100
19	扬佳（YZJM）	72712.00	249
40	雪花	66920.00	220
28	樱花（SAKURA）	51220.00	140
3	上菱	48856.00	62
38	长城（GREAT WALL FRIDGE）	42812.00	139
39	长虹（CHANGHONG）	37005.00	35

	brandname	total_actual_pay	sale_qtty
20	扬子（YANGZ）	32894.00	106
15	小米（MI）	28587.00	13
16	小鸭牌	24327.00	79
26	格兰仕（Galanz）	19301.00	9
21	新飞	15446.00	42
1	万宝（Wanbao）	15309.00	21
23	新飞（frestec）	11160.00	20

In [67]:

```
df8 = df7.copy()

df8 = df8[df8['user_site_province_id'] == '1'] # 省份取北京，数字是省份id

brand_sale_df_bj = df8.groupby('brandname', as_index=False).agg({'total_actual_pay':'sum', 'sale_qtty':'sum'}).sort_values(by='total_actual_pay', ascending=False)
brand_sale_df_bj = brand_sale_df_bj[(brand_sale_df_bj['brandname'] == '海尔（Haier）')|(brand_sale_df_bj['brandname'] == '容声（Ronshen）')|(brand_sale_df_bj['brandname'] == '西门子（SIEMENS）')|(brand_sale_df_bj['brandname'] == '美的（Midea）')]
brand_sale_df_bj
```

Out[67]:

	brandname	total_actual_pay	sale_qtty
26	海尔（Haier）	820296.00	325
33	西门子（SIEMENS）	413613.00	79
12	容声（Ronshen）	276533.00	122
31	美的（Midea）	100864.14	63

In [68]:

```
df8 = df7.copy()

df8 = df8[df8['brandname'] == '海尔（Haier）']

brand_sale_df_haier = df8.groupby('user_site_province_id', as_index=False).agg({'total_actual_pay':'sum', 'sale_qtty':'sum'}).sort_values(by='total_actual_pay', ascending=False)
brand_sale_df_haier = brand_sale_df_haier[(brand_sale_df_haier['user_site_province_id'] == '1')|(brand_sale_df_haier['user_site_province_id'] == '2')|(brand_sale_df_haier['user_site_province_id'] == '12')|(brand_sale_df_haier['user_site_province_id'] == '22')|(brand_sale_df_haier['user_site_province_id'] == '19')]
brand_sale_df_haier['user_site_province_id'] = brand_sale_df_haier['user_site_province_id'].astype('int')
brand_sale_df_haier.columns = ['province_id','total_actual_pay', 'sale_qtty']
brand_sale_df_haier.sort_values(by='province_id')
```

Out[68]:

	province_id	total_actual_pay	sale_qtty
1	1	820296.00	325
12	2	316005.00	139
4	12	422743.00	164
11	19	826869.01	366
15	22	291425.00	124

In [69]:

```
cust_price_df
```

Out[69]:

	province_id	total_pay	dim_province_name	order_num	user_num	cust_price
3	2	1425442.00	上海	592	538	2649.520446
5	13	938661.01	山东	394	383	2450.812037
2	12	1603704.00	江苏	714	704	2277.988636
1	1	2548956.74	北京	1180	1135	2245.776863
6	15	937725.00	浙江	444	422	2222.097156
7	17	712541.00	湖北	331	327	2179.024465
4	22	1104843.00	四川	557	548	2016.136861
9	27	564524.00	陕西	289	287	1966.982578
0	19	3547611.01	广东	1831	1813	1956.762830
11	7	499252.64	河南	294	285	1751.763649

In [70]:

```
order_num_df = cust_price_df[['province_id', 'order_num']][(cust_price_df['province_id'] == 1)|(cust_price_df['province_id'] == 12)|(cust_price_df['province_id'] == 19)|(cust_price_df['province_id'] == 2)|(cust_price_df['province_id'] == 22)]
order_num_df = order_num_df.sort_values(by='province_id')
order_num_df
```

Out[70]:

	province_id	order_num
1	1	1180
3	2	592
2	12	714
0	19	1831
4	22	557

In [71]:

```
brand_sale_df_haier = pd.merge(brand_sale_df_haier, order_num_df, on='province_id')
brand_sale_df_haier['渗透率'] = brand_sale_df_haier['sale_qtty'] / brand_sale_df_haier['order_num']
brand_sale_df_haier
```

Out[71]:

	province_id	total_actual_pay	sale_qtty	order_num	渗透率
0	19	826869.01	366	1831	0.199891
1	1	820296.00	325	1180	0.275424
2	12	422743.00	164	714	0.229692
3	2	316005.00	139	592	0.234797
4	22	291425.00	124	557	0.222621

In [72]:

```
def province_shentou(df, brandname, cust_price_df):
    df = df[df['brandname'] == brandname]

    brand_sale_df = df.groupby('user_site_province_id', as_index=False).agg({'total_actual_pay':'sum', 'sale_qtty':'sum'}).sort_values(by='total_actual_pay', ascending=False)
    brand_sale_df = brand_sale_df[(brand_sale_df['user_site_province_id'] == '1')|(brand_sale_df['user_site_province_id'] == '2')|(brand_sale_df['user_site_province_id'] == '12')|(brand_sale_df['user_site_province_id'] == '22')|(brand_sale_df['user_site_province_id'] == '19')]
    brand_sale_df['user_site_province_id'] = brand_sale_df['user_site_province_id'].astype('int')
    brand_sale_df.columns = ['province_id','total_actual_pay', 'sale_qtty']
    brand_sale_df.sort_values(by='province_id')

    order_num = cust_price_df[['province_id', 'order_num']][(cust_price_df['province_id'] == 1)|(cust_price_df['province_id'] == 12)|(cust_price_df['province_id'] == 19)|(cust_price_df['province_id'] == 2)|(cust_price_df['province_id'] == 22)]
    order_num = order_num.sort_values(by='province_id')

    brand_sale_df = pd.merge(brand_sale_df, order_num_df, on='province_id')
    brand_sale_df['渗透率'] = brand_sale_df['sale_qtty'] / brand_sale_df['order_num']
    brand_sale_df = brand_sale_df.sort_values(by='province_id')

    return brand_sale_df
```

In [73]:

```
df9 = df7.copy()

brand_sale_df_rs = province_shentou(df9, '容声（Ronshen）', cust_price_df)
brand_sale_df_siem = province_shentou(df9, '西门子（SIEMENS）', cust_price_df)
brand_sale_df_mi = province_shentou(df9, '美的（Midea）', cust_price_df)

brand_sale_df_siem
```

Out[73]:

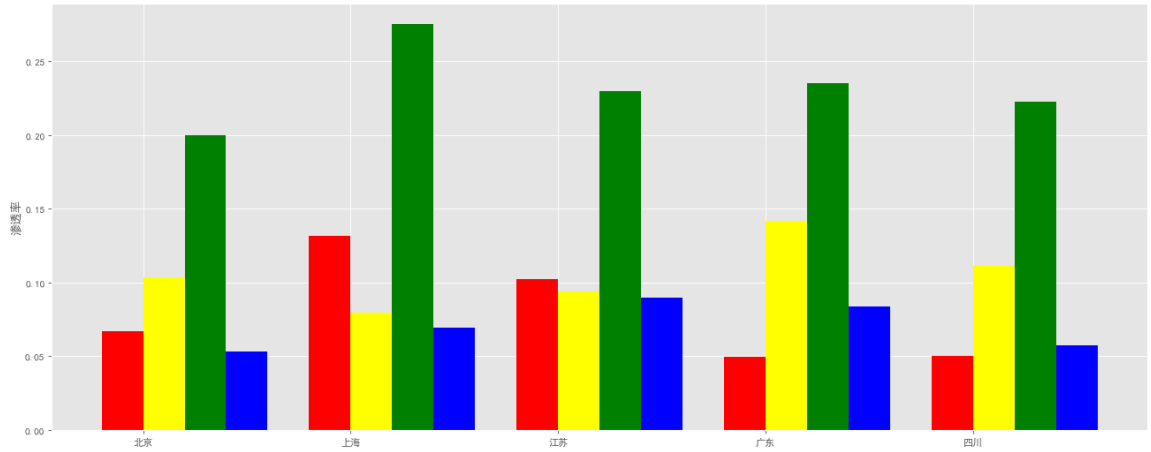
	province_id	total_actual_pay	sale_qtty	order_num	渗透率
1	1	413613.0	79	1180	0.066949
2	2	411564.0	78	592	0.131757
3	12	379749.0	73	714	0.102241
0	19	493163.0	91	1831	0.049700
4	22	159770.0	28	557	0.050269

In [74]:

```
plt.style.use('ggplot')

x = np.arange(5)

y1 = brand_sale_df_siem['渗透率']
y2 = brand_sale_df_rs['渗透率']
y3 = brand_sale_df_haier['渗透率']
y4 = brand_sale_df_mi['渗透率']

tick_label=['北京', '上海', '江苏', '广东', '四川']

total_width, n = 0.8, 4
width = total_width / n
x = x - (total_width - width) / 2

plt.figure(figsize=(20,8),dpi=80)
plt.style.use('ggplot')
plt.ylabel("渗透率")

bar_width = 0.2
plt.bar(x, y1, width=bar_width, color=['red'])
plt.bar(x+width, y2, width=bar_width, color=['yellow'])
plt.bar(x+2*width, y3, width=bar_width, color=['green'])
plt.bar(x+3*width, y4, width=bar_width, color=['blue'])

plt.xticks(x+bar_width/2, tick_label) # 显示x坐标轴的标签,即tick_label,调整位置，使其落在两个直方图中间位置
```

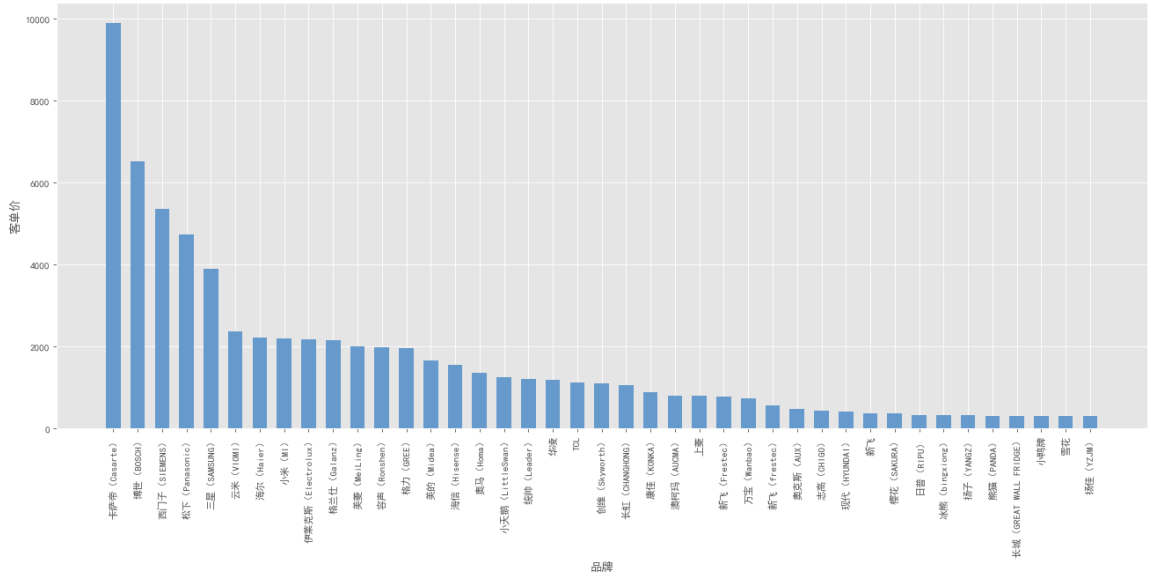
Out[74]:

```
([<matplotlib.axis.XTick at 0x203bfcce0b8>,
  <matplotlib.axis.XTick at 0x203bffb74e0>,
  <matplotlib.axis.XTick at 0x203bff15278>,
  <matplotlib.axis.XTick at 0x203bfd0b080>,
  <matplotlib.axis.XTick at 0x203bfcab470>],
 <a list of 5 Text xticklabel objects>)
```



In [75]:

```
plt.style.use('ggplot')

brand_sale_df['单价'] = brand_sale_df['total_actual_pay'] / brand_sale_df['sale_qtty']
brand_sale_df = brand_sale_df.sort_values(by='单价', ascending=False)

x = brand_sale_df['brandname']
y = brand_sale_df['单价']

plt.figure(figsize=(20,8),dpi=80)
plt.style.use('ggplot')
plt.xlabel('品牌')
plt.ylabel("客单价")

plt.xticks(range(len(x)), x, rotation=90)
rect = plt.bar(x, y, width=0.6, color=['#6699CC'])

plt.show()
```



In [ ]:

```

```