互评作业1: 数据探索性分析与预处理

对性别的分析

In [1]:
```
%%time
import pandas as pd
import pyarrow.parquet as pq
import numpy as np
import os
from collections import defaultdict
from pathlib import Path

data_type = "10G"
data_dir = Path(f"./data/{data_type}_data/")
file_paths = [file for file in data_dir.glob("*.parquet")]

gender_counts = defaultdict(int)

for file_path in file_paths:
    # 分块读取文件
    parquet_file = pq.ParquetFile(file_path)
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['gender'])
        df = table.to_pandas()

        chunk_counts = df['gender'].value_counts(dropna=False)
        for gender, count in chunk_counts.items():
            gender_counts[gender] += count

result_df = pd.DataFrame(list(gender_counts.items()), columns=['gender', 'count'
result_df = result_df.sort_values(by='count', ascending=False)
result_df = result_df.reset_index(drop=True)

result_df
```

```
CPU times: total: 5 s
Wall time: 5.4 s
```

Out[1]:

	gender	count
0	男	21603397
1	女	21598086
2	未指定	899652
3	其他	898865

男女比例近似1：1

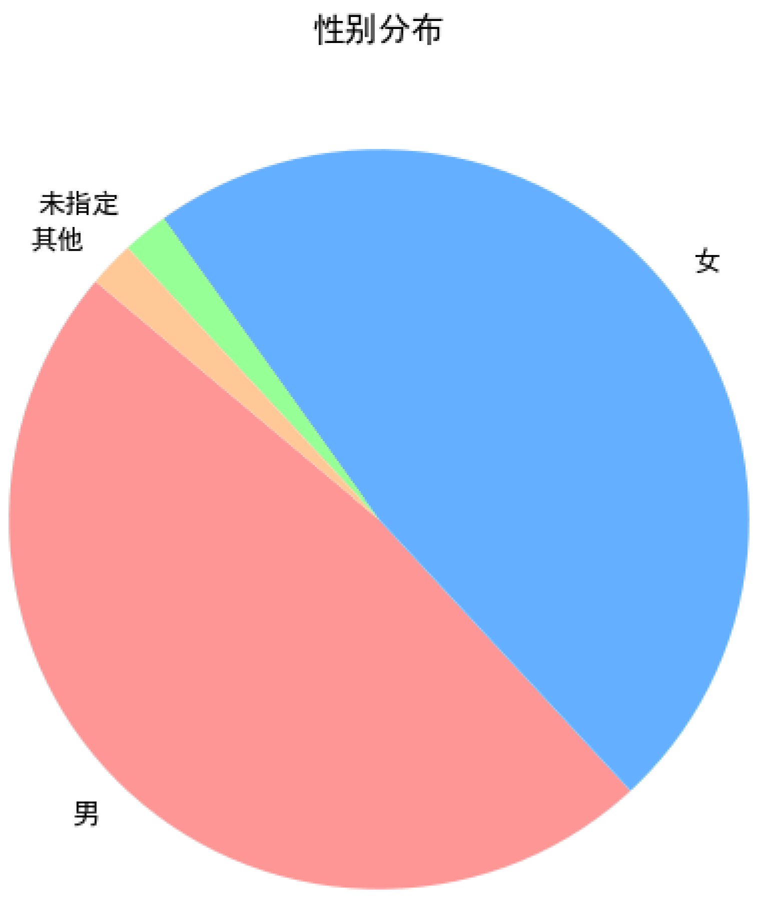
绘制饼状图

In [2]:
```
%%time
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

labels = result_df['gender']
sizes = result_df['count'].values

plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, startangle=140, colors=['#ff9999','#66b3ff','#99ff
plt.title('性别分布')

plt.show()
```



```
CPU times: total: 703 ms
Wall time: 759 ms
```

In [3]:
```
%%time
total_count = result_df['count'].sum()
error_value = result_df[result_df['gender'].isin(['未指定', '其他'])]['count'].s
print(f"总记录数: {total_count}")
print(f"性别异常值: {error_value}")
print(f"异常值占比: {(error_value/total_count*100):.2f}%")
```

```
总记录数: 45000000
性别异常值: 1798517
异常值占比: 4.00%
CPU times: total: 0 ns
Wall time: 1.34 ms
```

认为性别为“男”和“女”，为正常值，“未指定”和“其他”均为异常值

对国家的分析

In [4]:
```
%%time
country_counts = defaultdict(int)

for file_path in file_paths:
    parquet_file = pq.ParquetFile(file_path)

    # 分块读取文件
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['country'])
        df = table.to_pandas()

        chunk_counts = df['country'].value_counts(dropna=False)
        for country, count in chunk_counts.items():
            country_counts[country] += count

result_df = pd.DataFrame(list(country_counts.items()), columns=['country', 'coun
result_df = result_df.sort_values(by='count', ascending=False)
result_df = result_df.reset_index(drop=True)

result_df
```

```
CPU times: total: 4.67 s
Wall time: 4.72 s
```

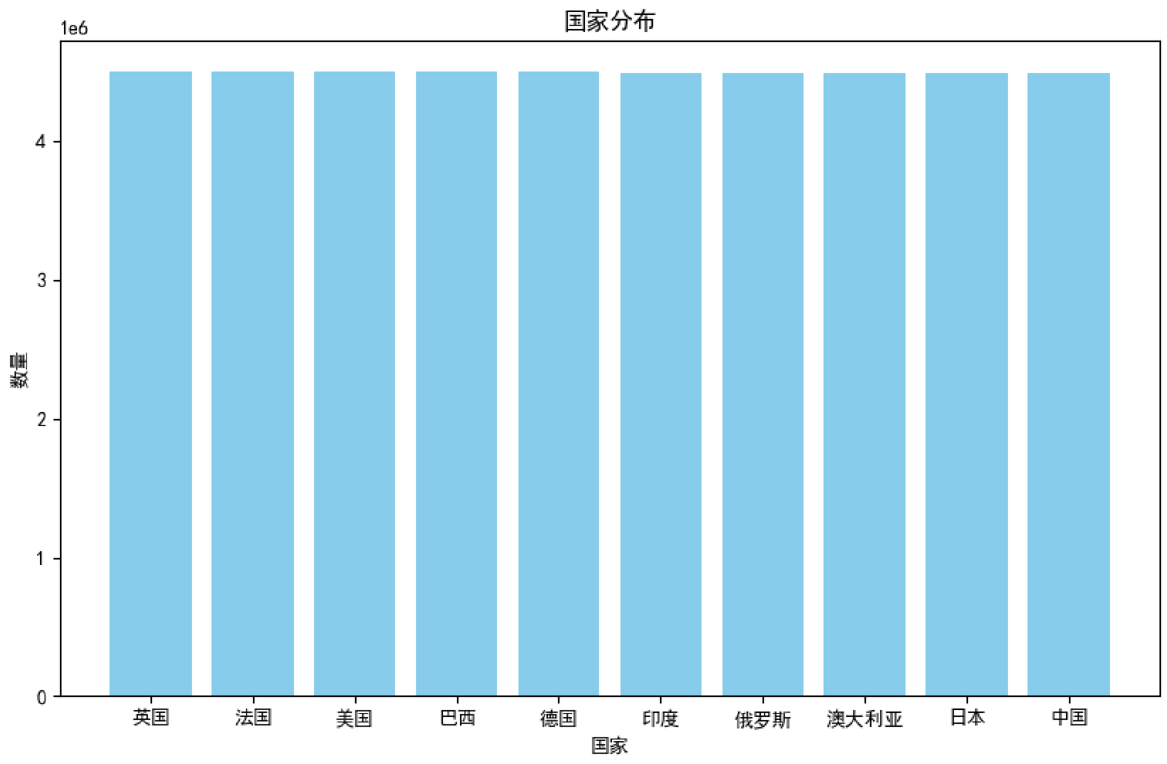
Out[4]:

	country	count
0	英国	4501669
1	法国	4501427
2	美国	4501158
3	巴西	4500526
4	德国	4500370
5	印度	4499562
6	俄罗斯	4499132
7	澳大利亚	4499124
8	日本	4498695
9	中国	4498337

绘制柱状图

In [5]:
```
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.figure(figsize=(10, 6))
plt.bar(result_df['country'], result_df['count'], color='skyblue')
plt.xlabel('国家')
plt.ylabel('数量')
plt.title('国家分布')
plt.show()
```



国家分布比较均匀

对年龄的分析

In [6]:
```
%%time
age_counts = defaultdict(int)

for file_path in file_paths:
    # 分块读取文件
    parquet_file = pq.ParquetFile(file_path)
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['age'])
        df = table.to_pandas()
        chunk_counts = df['age'].value_counts(dropna=False)
        for age, count in chunk_counts.items():
            age_counts[age] += count

result_df = pd.DataFrame(list(age_counts.items()), columns=['age', 'count'])
result_df = result_df.sort_values(by='age', ascending=True)
result_df = result_df.reset_index(drop=True)

result_df
```

```
CPU times: total: 734 ms
Wall time: 792 ms
```

Out[6]:

	age	count
0	18	542324
1	19	541384
2	20	542862
3	21	542716
4	22	542482
...	...	...
78	96	542174
79	97	542426
80	98	541367
81	99	542534
82	100	541845

83 rows × 2 columns

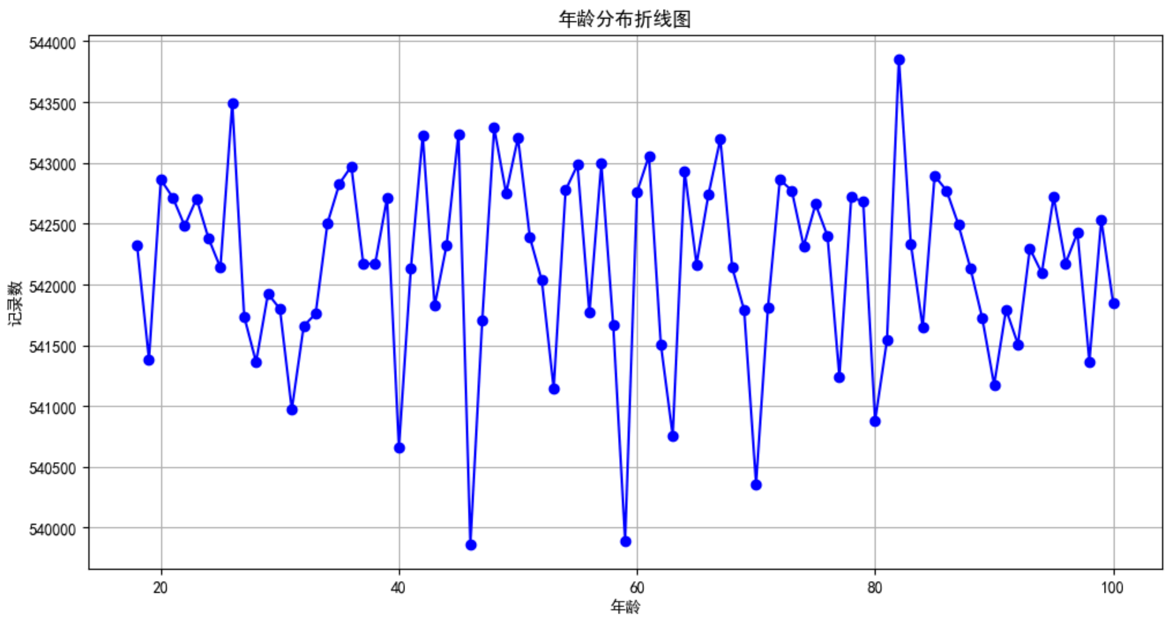
绘制折线图

In [7]:

```
%%time
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(result_df['age'], result_df['count'], marker='o', linestyle='-', color=
plt.xlabel('年龄')
plt.ylabel('记录数')
plt.title('年龄分布折线图')
plt.grid(True)

plt.show()
```



```
CPU times: total: 188 ms
Wall time: 191 ms
```

年龄分布比较均匀

对收入的分析

In [8]:
```
%%time
income_counts = defaultdict(int)
income_median = []

for file_path in file_paths:
    # 分块读取文件
    parquet_file = pq.ParquetFile(file_path)
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['income'])
        df = table.to_pandas()

        # 将收入按每1000分档
        bins = range(0, int(df['income'].max()) + 1000, 1000)
        df['income_bin'] = pd.cut(df['income'], bins=bins, right=False)

        # 统计每个分档的出现次数
        chunk_counts = df['income_bin'].value_counts(dropna=False)
        for income_bin, count in chunk_counts.items():
            income_counts[income_bin] += count

        for bin_interval in bins[:-1]:
            bin_median = (bin_interval + bin_interval + 1000) / 2
            income_median.append(bin_median)

# 将结果转换为 DataFrame
result_df = pd.DataFrame(list(income_counts.items()), columns=['income_bin', 'cc

# 按收入分档排序
result_df = result_df.sort_values(by='income_bin', ascending=True).reset_index(d

# 添加中位数列
result_df['income_median'] = income_median[:len(result_df)]

# 输出结果
result_df
```

```
CPU times: total: 5.86 s
Wall time: 6.02 s
```

Out[8]:

	income_bin	count	income_median
0	[0, 1000)	44870	500.0
1	[1000, 2000)	45287	1500.0
2	[2000, 3000)	44863	2500.0
3	[3000, 4000)	44989	3500.0
4	[4000, 5000)	45032	4500.0
...	...	...	...
995	[995000, 996000)	44845	995500.0
996	[996000, 997000)	45347	996500.0
997	[997000, 998000)	45088	997500.0
998	[998000, 999000)	44870	998500.0
999	[999000, 1000000)	45547	999500.0

1000 rows × 3 columns

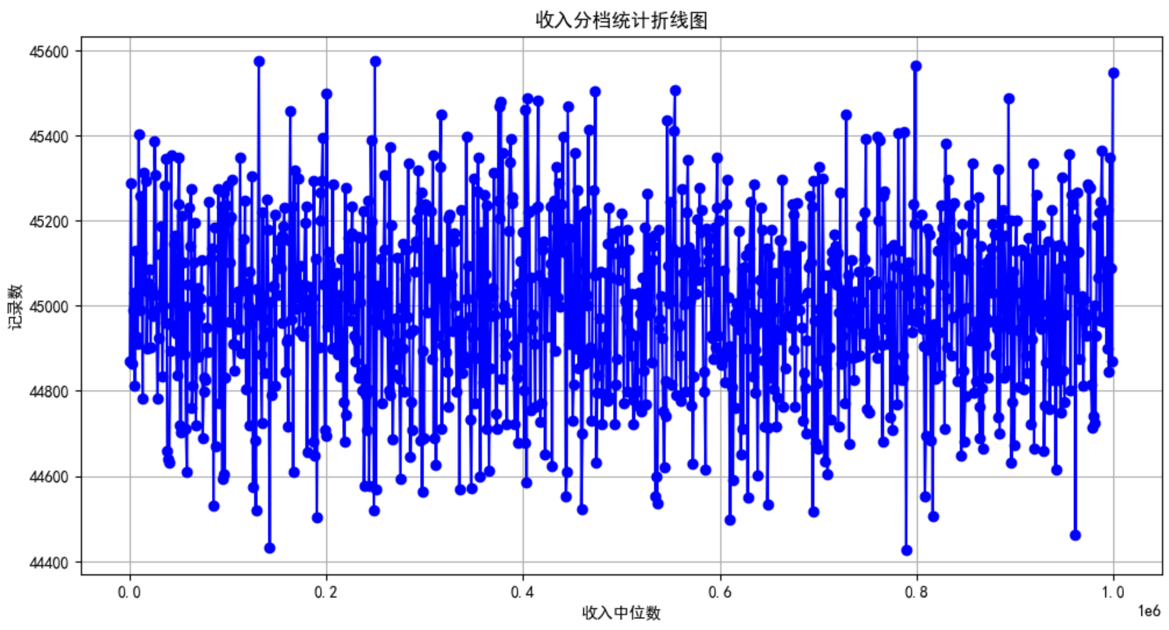
In [9]:
```python
import matplotlib.pyplot as plt

# 绘制折线图
plt.figure(figsize=(12, 6))
plt.plot(result_df['income_median'], result_df['count'], marker='o', linestyle='

# 设置坐标轴标签
plt.xlabel('收入中位数')
plt.ylabel('记录数')
plt.title('收入分档统计折线图')

# 显示网格
plt.grid(True)

# 显示图形
plt.show()
```



收入按1000分档，从图中可以看出收入分布也比较均匀

识别潜在高价值用户

In [10]:
```
%%time
import glob
# 筛选年龄在20——35岁之间，收入50000以上的用户作为潜在高价值用户
# 新开一列表示该用户是否为高价值用户，1表示该用户是高价值用户，0则不是
high_counts = defaultdict(int)
for index, file_path in enumerate(file_paths):
    parquet_file = pq.ParquetFile(file_path)
    print(parquet_file)

    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i)
        df = table.to_pandas()
        conditions = (
            (df['age'].between(20, 35)) &
            (df['income'] > 50000)
        )
        df['is_high_value_user'] = conditions.astype(int)
        temp_file_path = f"part-{index}-temp_chunk_{i}.parquet"
        df.to_parquet(temp_file_path)
        chunk_counts = df['is_high_value_user'].value_counts(dropna=False)
        for high, count in chunk_counts.items():
            high_counts[high] += count

    temp_files = glob.glob(f"part-{index}-temp_chunk_*.parquet")
    merged_df = pd.concat([pd.read_parquet(file) for file in temp_files], ignore
    save_path = f"result/{data_type}_data/part-0000{index}.parquet"
    merged_df.to_parquet(save_path)
    merged_df.head()
    # 删除临时文件
    for file in temp_files:
        os.remove(file)

result_df = pd.DataFrame(list(high_counts.items()), columns=['high_counts', 'cou
# result_df = result_df.sort_values(by='count', ascending=False)
# result_df = result_df.reset_index(drop=True)

result_df

total_count = result_df['count'].sum()
high_count = result_df[result_df['high_counts'].isin([1])]['count'].sum()
print(f"总记录数: {total_count}")
print(f"高价值用户数: {high_count}")
print(f"占比: {(high_count/total_count*100):.2f}%")
```

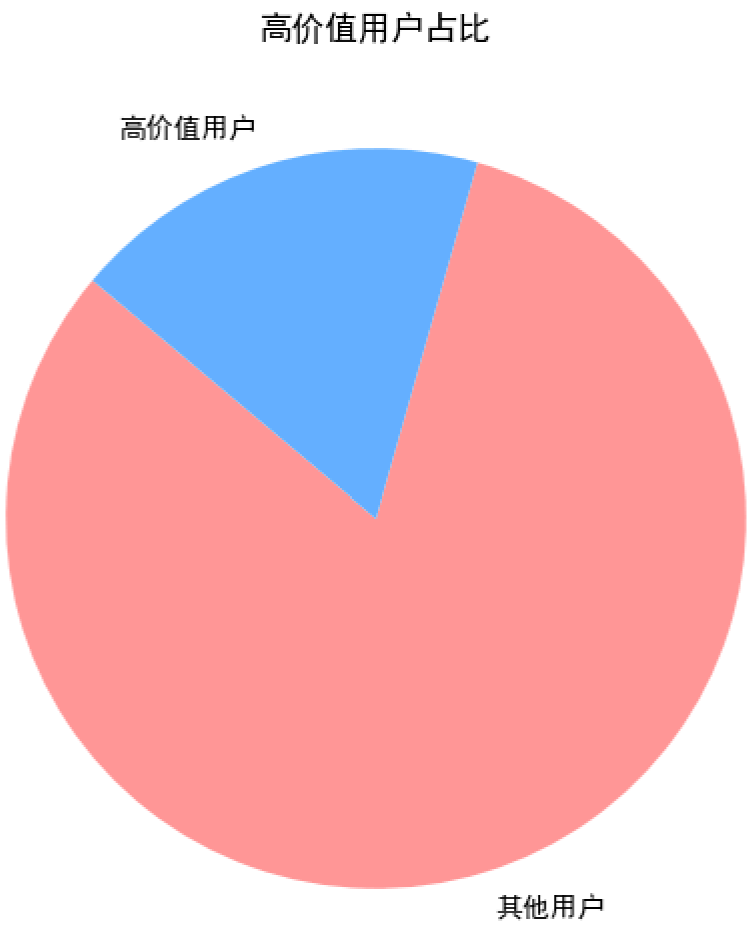
```
<pyarrow.parquet.core.ParquetFile object at 0x0000025D9A067B10>
<pyarrow.parquet.core.ParquetFile object at 0x0000025D99894E90>
<pyarrow.parquet.core.ParquetFile object at 0x0000025E86D4DBD0>
<pyarrow.parquet.core.ParquetFile object at 0x000002609CF2B6D0>
<pyarrow.parquet.core.ParquetFile object at 0x0000025F3F86EC10>
<pyarrow.parquet.core.ParquetFile object at 0x000002606C9153D0>
<pyarrow.parquet.core.ParquetFile object at 0x0000025E6D0A6250>
<pyarrow.parquet.core.ParquetFile object at 0x000002608050AB10>
总记录数: 45000000
高价值用户数: 8240509
占比: 18.31%
CPU times: total: 22min 12s
Wall time: 22min 13s
```

In [11]:
```
%%time
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# labels = result_df['high_counts']
labels = ["其他用户", "高价值用户"]
sizes = result_df['count'].values

plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, startangle=140, colors=['#ff9999','#66b3ff','#99ff
plt.title('高价值用户占比')

plt.show()
```



```
CPU times: total: 125 ms
Wall time: 200 ms
```

互评作业1: 数据探索性分析与预处理

对性别的分析

In [11]:
```
%%time
import pandas as pd
import pyarrow.parquet as pq
import numpy as np
import os
from collections import defaultdict
from pathlib import Path

data_type = "30G"
data_dir = Path(f"./data/{data_type}_data/")
file_paths = [file for file in data_dir.glob("*.parquet")]

gender_counts = defaultdict(int)

for file_path in file_paths:
    # 分块读取文件
    parquet_file = pq.ParquetFile(file_path)
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['gender'])
        df = table.to_pandas()

        chunk_counts = df['gender'].value_counts(dropna=False)
        for gender, count in chunk_counts.items():
            gender_counts[gender] += count

result_df = pd.DataFrame(list(gender_counts.items()), columns=['gender', 'count'
result_df = result_df.sort_values(by='count', ascending=False)
result_df = result_df.reset_index(drop=True)

result_df
```

```
CPU times: total: 13 s
Wall time: 13.9 s
```

Out[11]:

	gender	count
0	男	64810501
1	女	64792563
2	未指定	2698564
3	其他	2698372

男女比例近似1：1

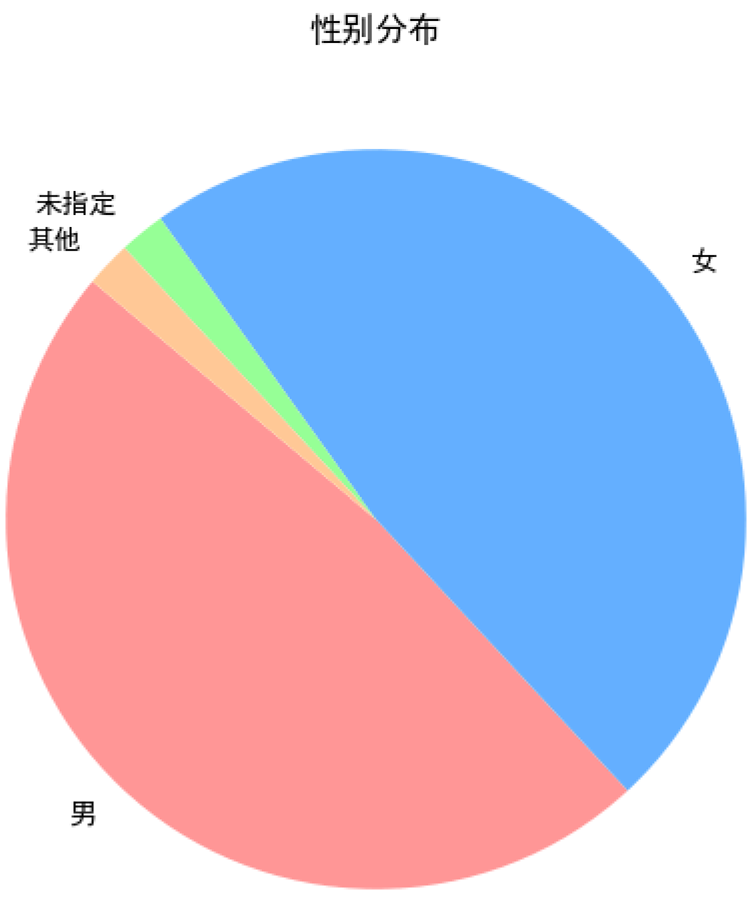
绘制饼状图

In [12]:
```
%%time
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

labels = result_df['gender']
sizes = result_df['count'].values

plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, startangle=140, colors=['#ff9999','#66b3ff','#99ff
plt.title('性别分布')

plt.show()
```



```
CPU times: total: 109 ms
Wall time: 192 ms
```

In [13]:

```
%%time
total_count = result_df['count'].sum()
error_value = result_df[result_df['gender'].isin(['未指定', '其他'])]['count'].s
print(f"总记录数: {total_count}")
print(f"性别异常值: {error_value}")
print(f"异常值占比: {(error_value/total_count*100):.2f}%")
```

```
总记录数: 135000000
性别异常值: 5396936
异常值占比: 4.00%
CPU times: total: 0 ns
Wall time: 13.6 ms
```

认为性别为“男”和“女”，为正常值，“未指定”和“其他”均为异常值

对国家的分析

In [14]:
```
%%time
country_counts = defaultdict(int)

for file_path in file_paths:
    parquet_file = pq.ParquetFile(file_path)

    # 分块读取文件
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['country'])
        df = table.to_pandas()

        chunk_counts = df['country'].value_counts(dropna=False)
        for country, count in chunk_counts.items():
            country_counts[country] += count

result_df = pd.DataFrame(list(country_counts.items()), columns=['country', 'coun
result_df = result_df.sort_values(by='count', ascending=False)
result_df = result_df.reset_index(drop=True)

result_df
```

```
CPU times: total: 17.5 s
Wall time: 18.2 s
```

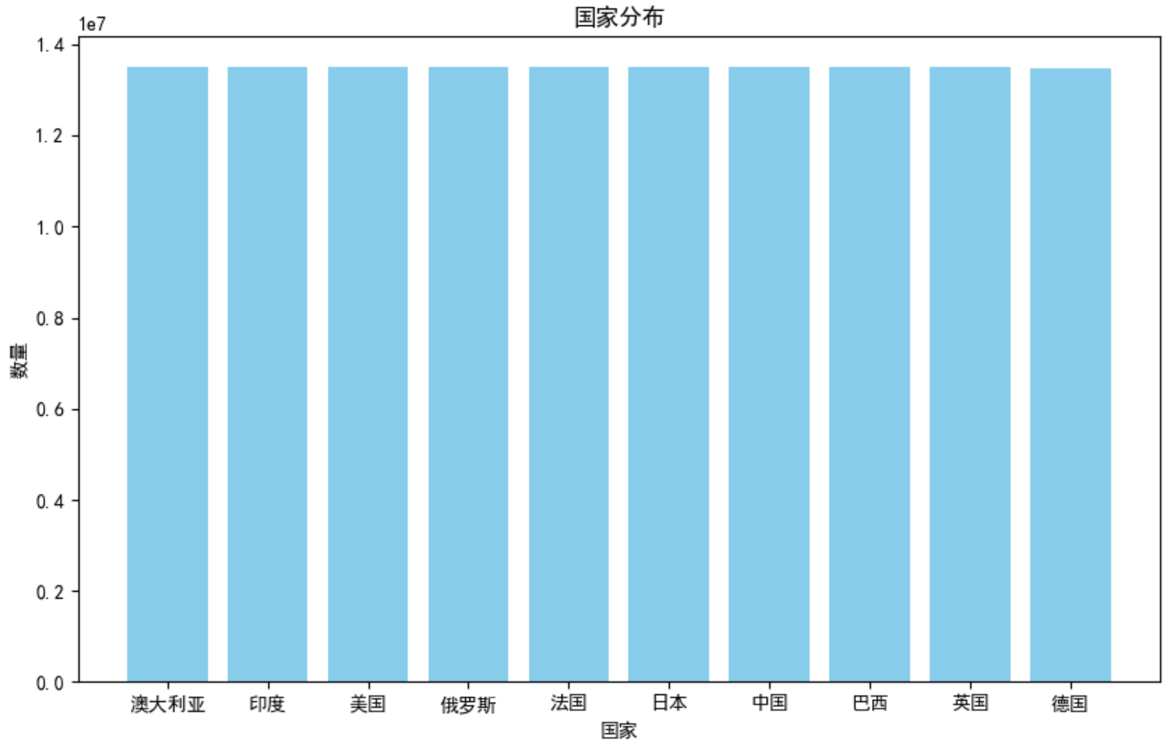
Out[14]:

	country	count
0	澳大利亚	13502953
1	印度	13502855
2	美国	13502589
3	俄罗斯	13500996
4	法国	13499078
5	日本	13498944
6	中国	13498904
7	巴西	13498665
8	英国	13498183
9	德国	13496833

绘制柱状图

In [15]:
```
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.figure(figsize=(10, 6))
plt.bar(result_df['country'], result_df['count'], color='skyblue')
plt.xlabel('国家')
plt.ylabel('数量')
plt.title('国家分布')
plt.show()
```



国家分布比较均匀

对年龄的分析

```
In [16]: %%time
age_counts = defaultdict(int)

for file_path in file_paths:
    # 分块读取文件
    parquet_file = pq.ParquetFile(file_path)
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['age'])
        df = table.to_pandas()
        chunk_counts = df['age'].value_counts(dropna=False)
        for age, count in chunk_counts.items():
            age_counts[age] += count

result_df = pd.DataFrame(list(age_counts.items()), columns=['age', 'count'])
result_df = result_df.sort_values(by='age', ascending=True)
result_df = result_df.reset_index(drop=True)

result_df
```

```
CPU times: total: 2.39 s
Wall time: 2.46 s
```

Out[16]:

	age	count
0	18	1626999
1	19	1627292
2	20	1625687
3	21	1625424
4	22	1626915
...	...	...
78	96	1626428
79	97	1626394
80	98	1625673
81	99	1626035
82	100	1629290

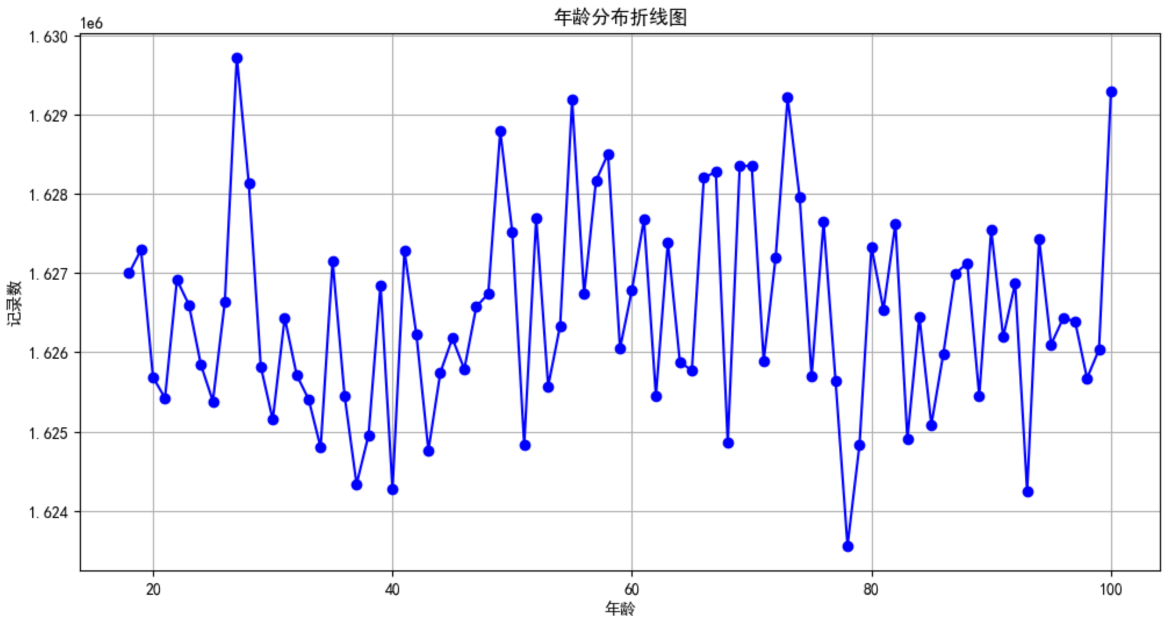
83 rows × 2 columns

绘制折线图

In [17]:
```
%%time
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(result_df['age'], result_df['count'], marker='o', linestyle='-', color=
plt.xlabel('年龄')
plt.ylabel('记录数')
plt.title('年龄分布折线图')
plt.grid(True)

plt.show()
```



```
CPU times: total: 172 ms
Wall time: 186 ms
```

年龄分布比较均匀

对收入的分析

In [ ]:
```
%%time
income_counts = defaultdict(int)
income_median = []

for file_path in file_paths:
    # 分块读取文件
    parquet_file = pq.ParquetFile(file_path)
    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i, columns=['income'])
        df = table.to_pandas()

        # 将收入按每1000分档
        bins = range(0, int(df['income'].max()) + 1000, 1000)

        # 删除里面的空值
        df['income_bin'] = pd.cut(df['income'], bins=bins, right=False)

        df.dropna(subset=['income_bin'], inplace=True)
        # 统计每个分档的出现次数
        chunk_counts = df['income_bin'].value_counts(dropna=False)
        for income_bin, count in chunk_counts.items():
            income_counts[income_bin] += count

        for bin_interval in bins[:-1]:
            bin_median = (bin_interval + bin_interval + 1000) / 2
            income_median.append(bin_median)

# 将结果转换为 DataFrame
result_df = pd.DataFrame(list(income_counts.items()), columns=['income_bin', 'cc

# 按收入分档排序
result_df = result_df.sort_values(by='income_bin', ascending=True).reset_index(d

# 添加中位数列
result_df['income_median'] = income_median[:len(result_df)]

# 输出结果
result_df
```

```
CPU times: total: 24 s
Wall time: 25 s
```

Out[ ]:

	income_bin	count	income_median
0	[0, 1000)	135091	500.0
1	[1000, 2000)	135008	1500.0
2	[2000, 3000)	135011	2500.0
3	[3000, 4000)	135475	3500.0
4	[4000, 5000)	135163	4500.0
...	...	...	...
995	[995000, 996000)	135050	995500.0
996	[996000, 997000)	135284	996500.0
997	[997000, 998000)	135188	997500.0
998	[998000, 999000)	135389	998500.0
999	[999000, 1000000)	135372	999500.0

1000 rows × 3 columns

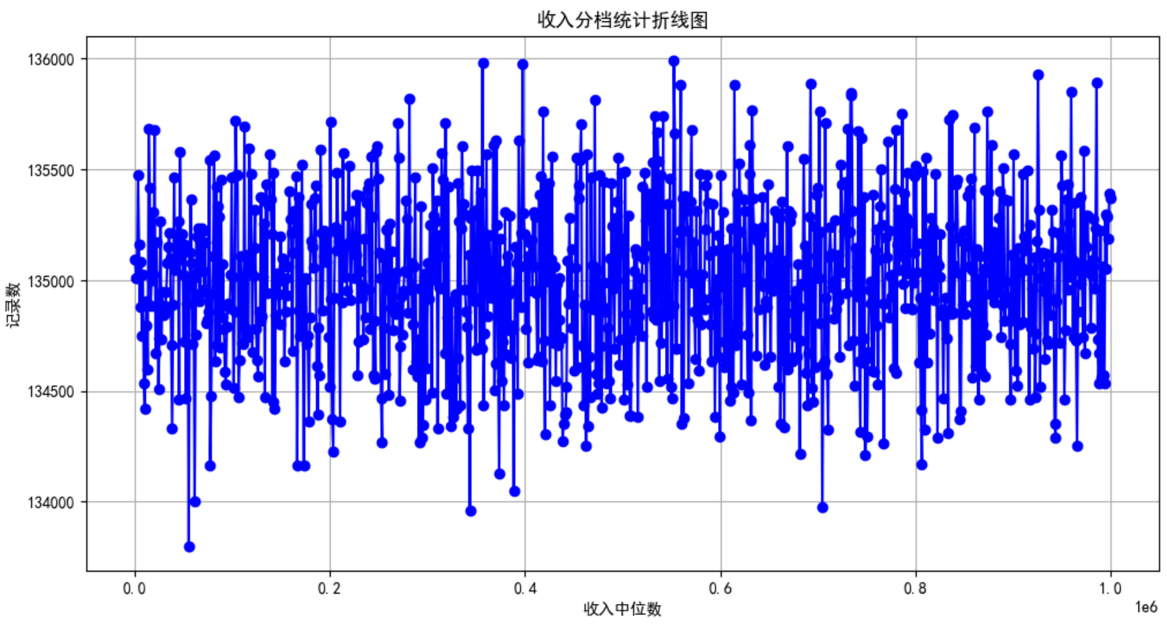
In [19]:
```python
import matplotlib.pyplot as plt

# 绘制折线图
plt.figure(figsize=(12, 6))
plt.plot(result_df['income_median'], result_df['count'], marker='o', linestyle='

# 设置坐标轴标签
plt.xlabel('收入中位数')
plt.ylabel('记录数')
plt.title('收入分档统计折线图')

# 显示网格
plt.grid(True)

# 显示图形
plt.show()
```



收入按1000分档，从图中可以看出收入分布也比较均匀

识别潜在高价值用户

In [20]:
```
%%time
import glob
import pandas as pd
import pyarrow.parquet as pq
import numpy as np
import os
from collections import defaultdict
from pathlib import Path

data_type = "30G"
data_dir = Path(f"./data/{data_type}_data/")
file_paths = [file for file in data_dir.glob("*.parquet")]
# 筛选年龄在20——35岁之间，收入50000以上的用户作为潜在高价值用户
# 新开一列表示该用户是否为高价值用户，1表示该用户是高价值用户，0则不是
high_counts = defaultdict(int)
for index, file_path in enumerate(file_paths):
    parquet_file = pq.ParquetFile(file_path)
    print(parquet_file)

    for i in range(parquet_file.num_row_groups):
        table = parquet_file.read_row_group(i)
        df = table.to_pandas()
        conditions = (
            (df['age'].between(20, 35)) &
            (df['income'] > 50000)
        )
        df['is_high_value_user'] = conditions.astype(int)
        temp_file_path = f"part-{index}-temp_chunk_{i}.parquet"
        df.to_parquet(temp_file_path)
        chunk_counts = df['is_high_value_user'].value_counts(dropna=False)
        for high, count in chunk_counts.items():
            high_counts[high] += count

    temp_files = glob.glob(f"part-{index}-temp_chunk_*.parquet")
    merged_df = pd.concat([pd.read_parquet(file) for file in temp_files], ignore
    save_path = f"result/{data_type}_data/part-0000{index}.parquet"
    merged_df.to_parquet(save_path)
    merged_df.head()
    # 删除临时文件
    for file in temp_files:
        os.remove(file)

result_df = pd.DataFrame(list(high_counts.items()), columns=['high_counts', 'cou
# result_df = result_df.sort_values(by='count', ascending=False)
# result_df = result_df.reset_index(drop=True)

result_df

total_count = result_df['count'].sum()
high_count = result_df[result_df['high_counts'].isin([1])]['count'].sum()
print(f"总记录数：{total_count}")
print(f"高价值用户数：{high_count}")
print(f"占比：{(high_count/total_count*100):.2f}%")
```

```
<pyarrow.parquet.core.ParquetFile object at 0x000001FBDB37E990>
<pyarrow.parquet.core.ParquetFile object at 0x000001FBD31907D0>
<pyarrow.parquet.core.ParquetFile object at 0x000001FBDB381D10>
<pyarrow.parquet.core.ParquetFile object at 0x0000020095814290>
<pyarrow.parquet.core.ParquetFile object at 0x000001FD3DEABA10>
<pyarrow.parquet.core.ParquetFile object at 0x0000020023EF0490>
<pyarrow.parquet.core.ParquetFile object at 0x000001FD80486410>
<pyarrow.parquet.core.ParquetFile object at 0x000001FFE11DE1D0>
<pyarrow.parquet.core.ParquetFile object at 0x000001FDB522E290>
<pyarrow.parquet.core.ParquetFile object at 0x000001FFECF58D10>
<pyarrow.parquet.core.ParquetFile object at 0x000001FE3EAAEA90>
<pyarrow.parquet.core.ParquetFile object at 0x0000020041EACA10>
<pyarrow.parquet.core.ParquetFile object at 0x000001FDFBFCDD90>
<pyarrow.parquet.core.ParquetFile object at 0x000002001D010290>
<pyarrow.parquet.core.ParquetFile object at 0x000001FFE6924450>
<pyarrow.parquet.core.ParquetFile object at 0x000002008E26C210>
总记录数: 135000000
高价值用户数: 24718719
占比: 18.31%
CPU times: total: 1h 4min
Wall time: 1h 26min 27s
```

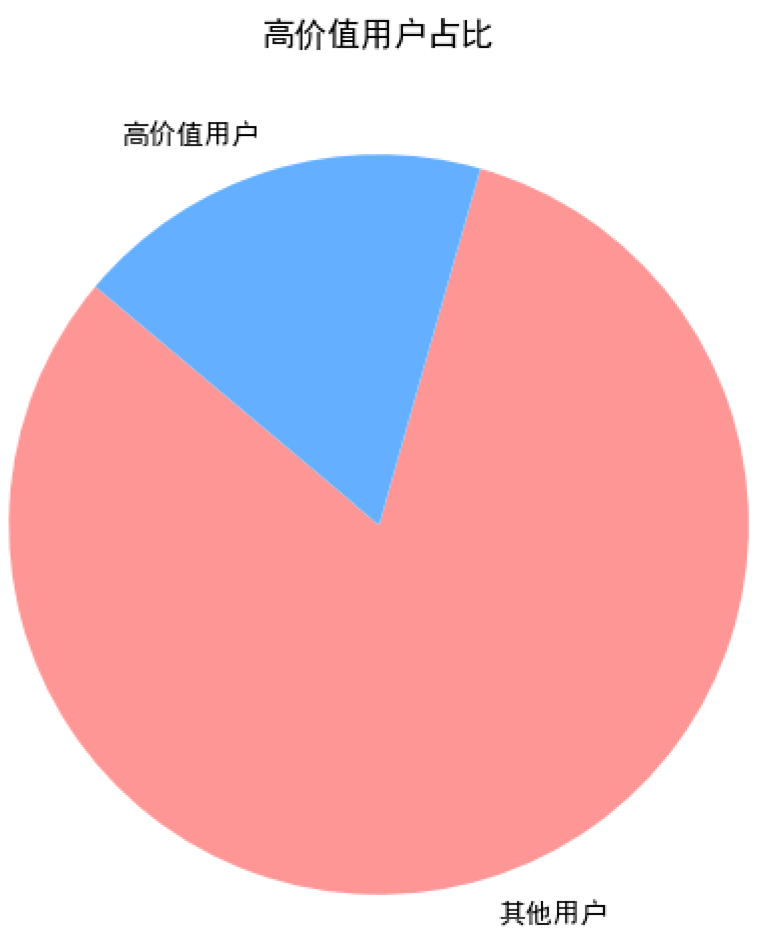
In [21]:

```
%%time
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# labels = result_df['high_counts']
labels = ["其他用户", "高价值用户"]
sizes = result_df['count'].values

plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, startangle=140, colors=['#ff9999','#66b3ff','#99ff
plt.title('高价值用户占比')

plt.show()
```



```
CPU times: total: 156 ms
Wall time: 373 ms
```