《大数据技术基础综合实验》

姓名：李志毅

班级：2018211314

学号：2018211582

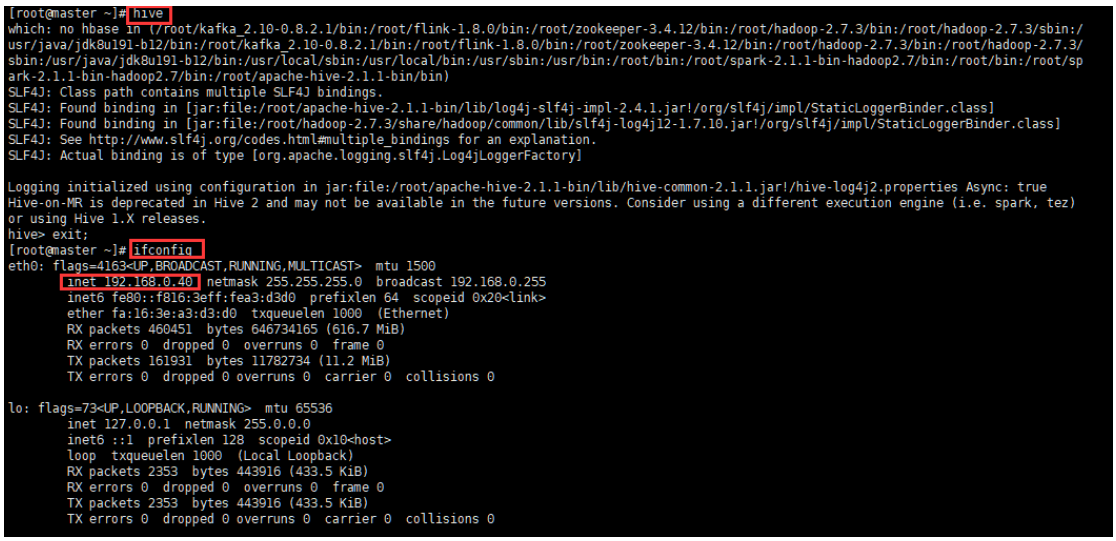
一、实验截图

实验截图 1

```
mysql> show variables like '%character%';
+--------------------------+--------------------------------+
| Variable_name            | Value                          |
+--------------------------+--------------------------------+
| character_set_client     | utf8mb3                        |
| character_set_connection | utf8mb3                        |
| character_set_database   | utf8mb3                        |
| character_set_filesystem | binary                         |
| character_set_results    | utf8mb3                        |
| character_set_server     | utf8mb3                        |
| character_set_system     | utf8mb3                        |
| character_sets_dir       | /usr/share/mysql-8.0/charsets/ |
+--------------------------+--------------------------------+
8 rows in set (0.01 sec)

mysql> quit;
Bye
[root@master ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.40  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fea3:d3d0  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:a3:d3:d0  txqueuelen 1000  (Ethernet)
        RX packets 336477  bytes 485071420 (462.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 117228  bytes 8261734 (7.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

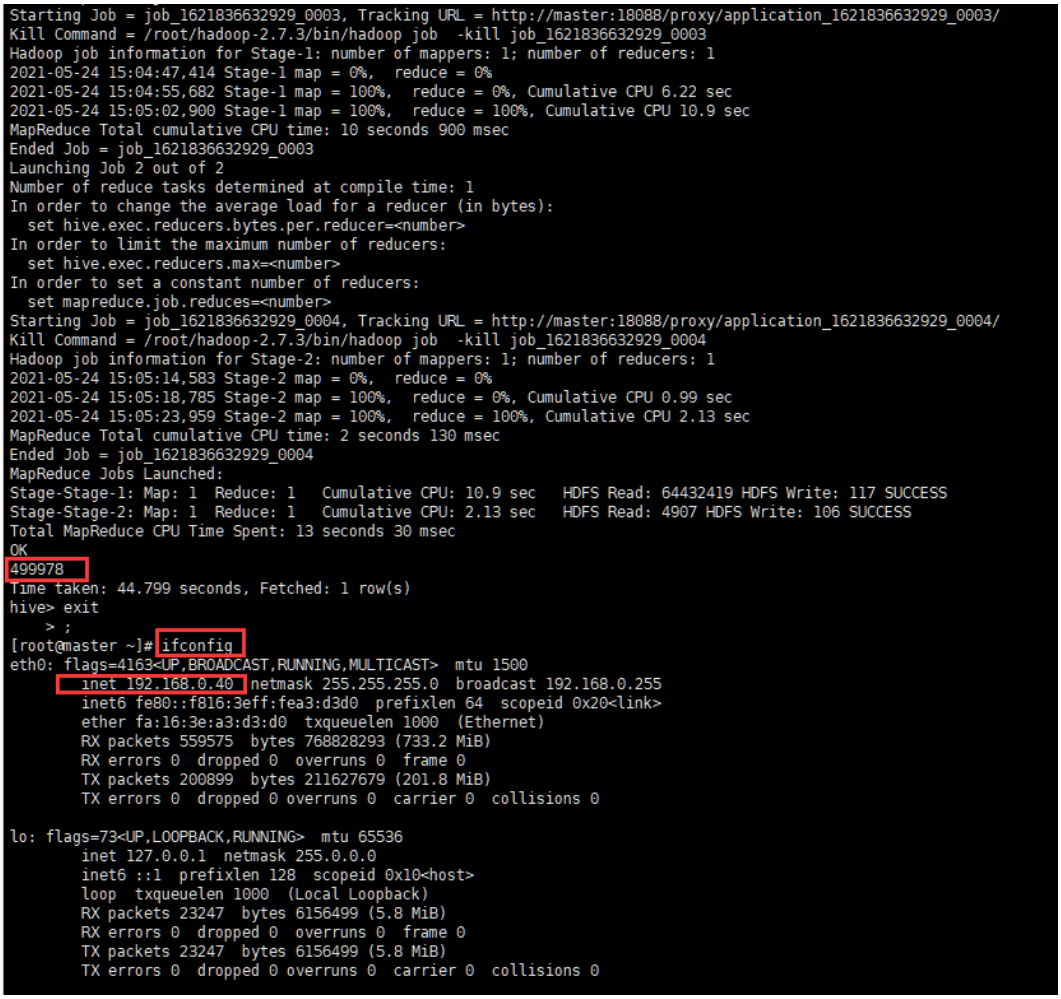
实验截图 1:查看 mysql 编码

实验截图 2



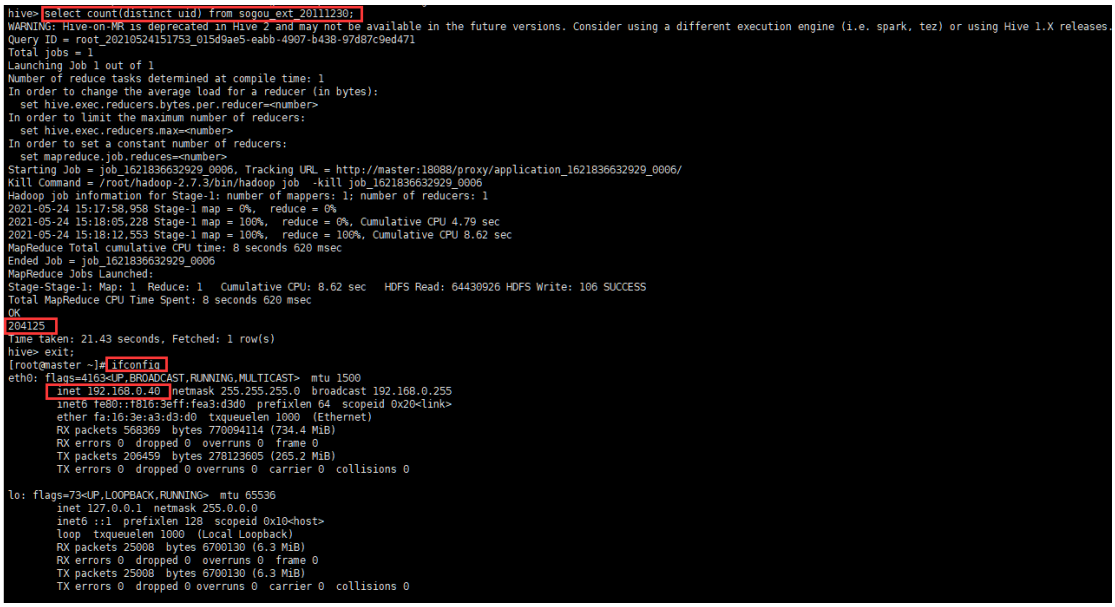
实验截图 2:hive 启动成功

实验截图 3



实验截图 3:无重复总条数

实验截图 4



实验截图 4:独立 UID 条数

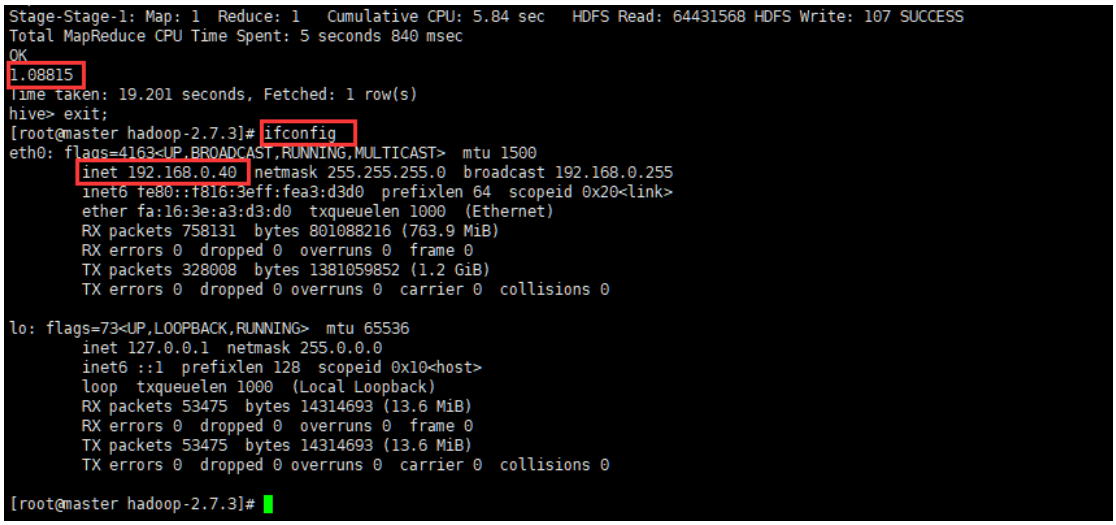
实验截图 5 第一种

```
select avg(length(keyword)) from sogou_ext_20111230;
```

实验截图 5:关键词平均长度统计

实验截图 5 第二种

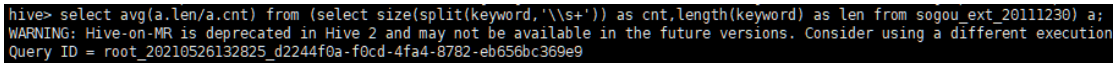
```
select avg(a.cnt) from (select size(split(keyword,'\\s+')) as cnt from sogou_ext_20111230) a;
```

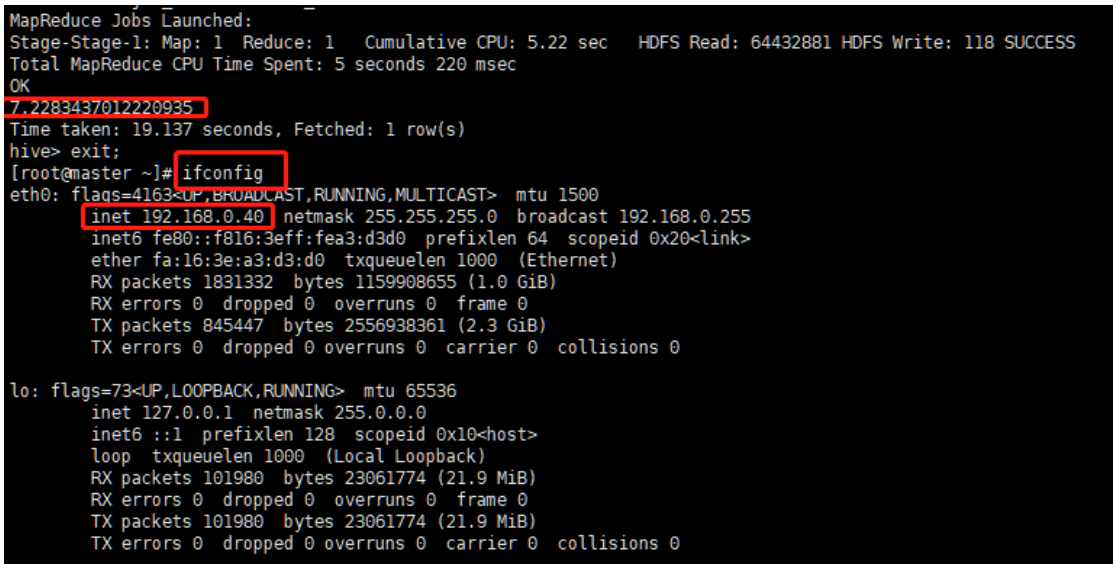


实验截图 5:关键词平均长度统计

实验截图 5 第三种

```
select avg(a.len/a.cnt) from (select size(split(keyword,'\\s+')) as cnt,length(split(keyword,'\\s+')) as len from sogou_ext_20111230) a;
```





实验截图 5:关键词平均长度统计

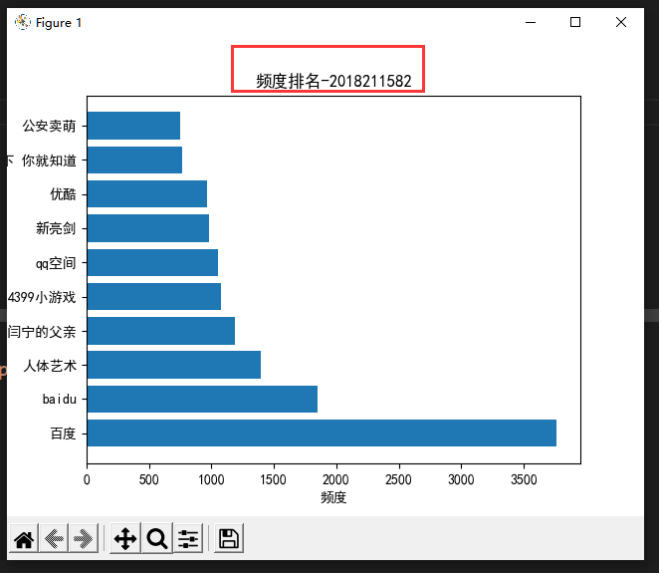
实验截图 6

```
hive> select count(*) from(select count(*) as query_count  from sogou_ext_20111230 group by uid having query_count > 2) a;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spa
rk, tez) or using Hive 1.X releases.
Query ID = root_20210524163701_676da900-6ca3-444d-9480-71f8967885e5
Starting Job = job_1621836632929_0010, Tracking URL = http://master:18088/proxy/application_1621836632929_0010/
Kill Command = /root/hadoop-2.7.3/bin/hadoop job  -kill job_1621836632929_0010
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2021-05-24 16:37:34,857 Stage-2 map = 0%,  reduce = 0%
2021-05-24 16:37:40,085 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.79 sec
2021-05-24 16:37:45,239 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.23 sec
MapReduce Total cumulative CPU time: 2 seconds 230 msec
Ended Job = job_1621836632929_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.36 sec   HDFS Read: 64430749 HDFS Write: 116
SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.23 sec   HDFS Read: 4901 HDFS Write: 105 SUCC
ESS
Total MapReduce CPU Time Spent: 10 seconds 590 msec
OK
56711
Time taken: 44.927 seconds, Fetched: 1 row(s)
hive> exit;
[root@master ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.40  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fea3:d3d0  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:a3:d3:d0  txqueuelen 1000  (Ethernet)
```

实验截图 6：查询次数大于 2 次的用户总数

实验截图 7

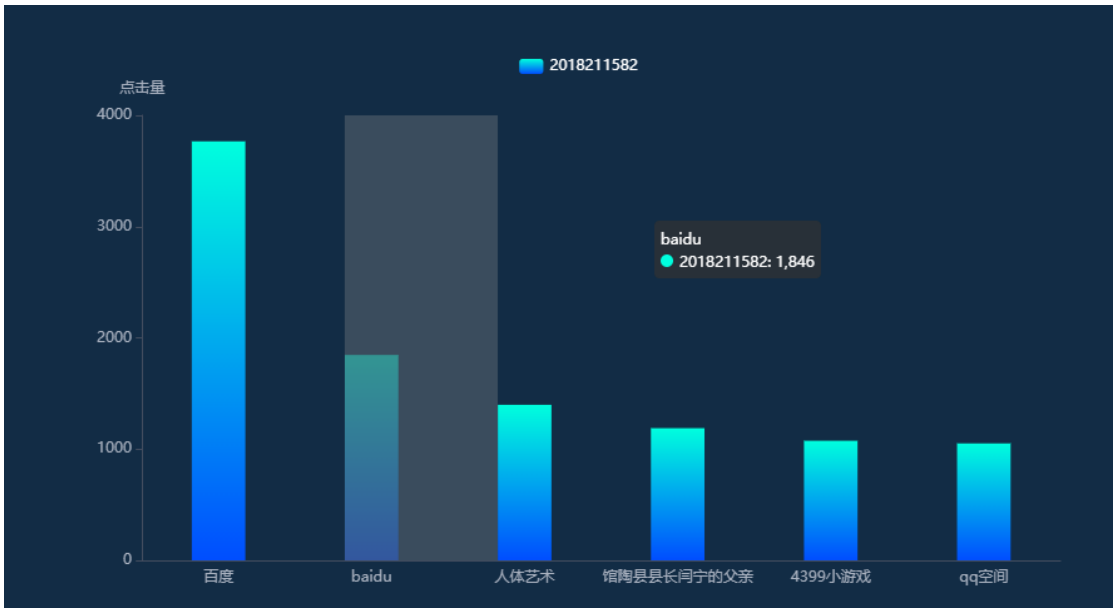
```
py1.py
E: > 大三下 > 带数据 > 期末综合实验 > py1.py > ...
1  from pyhive import hive
2  import matplotlib.pyplot as plt
3  plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）
4  plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）
5  conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')
6  cursor = conn.cursor()
7  cursor.execute('select keyword,count(*) as cnt from sogou_100w.sogou_ext_20111230 group by key
8  keywords = []
9  frequency = []
10 for result in cursor.fetchall():
11     keywords.append(result[0])
12     frequency.append(result[1])
13 cursor.close()
14 conn.close()
15 plt.barh(keywords, frequency)
16 plt.title('频度排名-2018211582')
17 plt.xlabel('频度')
18 plt.ylabel('关键词')
19 plt.show()

输出  终端  调试控制台  问题
[Running] set PYTHONIOENCODING=utf8 && p
```



实验截图 7：搜索前 10 的 Python 可视化

实验截图 8

```
OK
百度    3766
baidu   1846
人体艺术        1398
馆陶县县长闫宁的父亲    1188
4399小游戏      1075
qq空间  1051
新亮剑  979
优酷    966
百度一下 你就知道       762
公安卖萌        746
百度一下        724
4399    709
qq网名  609
魏特琳  566
```



实验截图 8：华为 DLV 可视化截图

二、实验分析

1. 分析 rank 与用户点击次数之间的关系

- **根据要求，设计 HiveSQL 语句：**
  根据 rank 进行分组，统计每组的总数，即为点击数，根据 rank 升序排序数据，因此 HiveSQL 语句如下：

```
SELECT rank,COUNT(*) AS cnt FROM sogou_100w.sogou_ext_20111230 GROUP BY rank O
RDER BY rank LIMIT 30
```

查询结果如下：

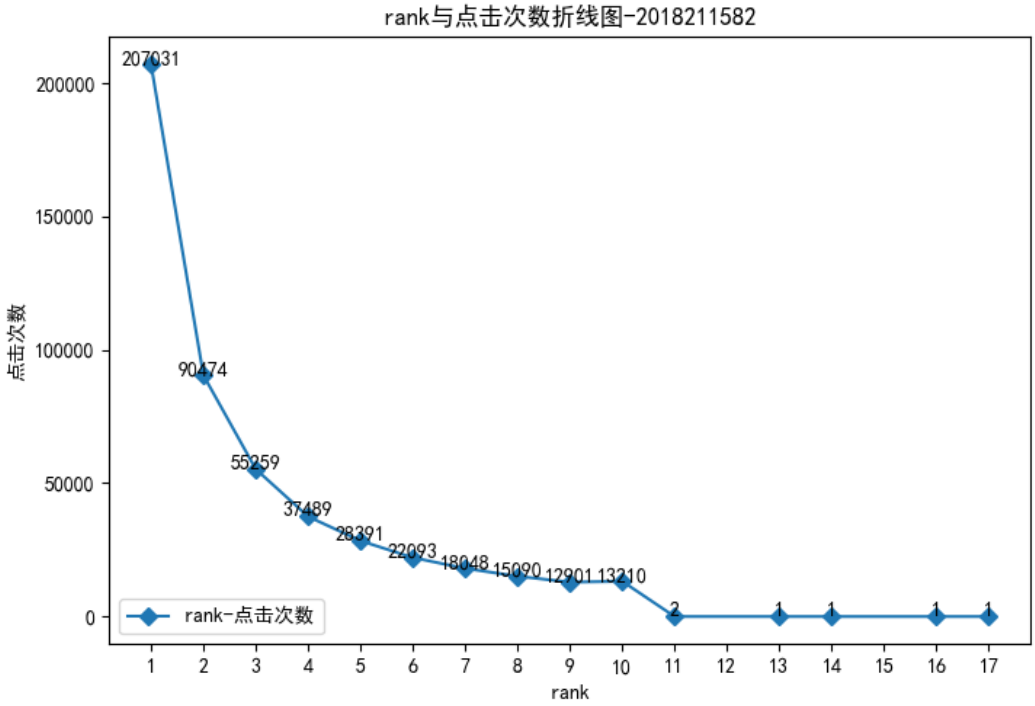
```
1       207031
2       90474
3       55259
4       37489
5       28391
6       22093
7       18048
8       15090
9       12901
10      13210
11      2
13      1
14      1
16      1
17      1
18      1
19      1
25      1
27      2
30      1
36      1
39      1
```

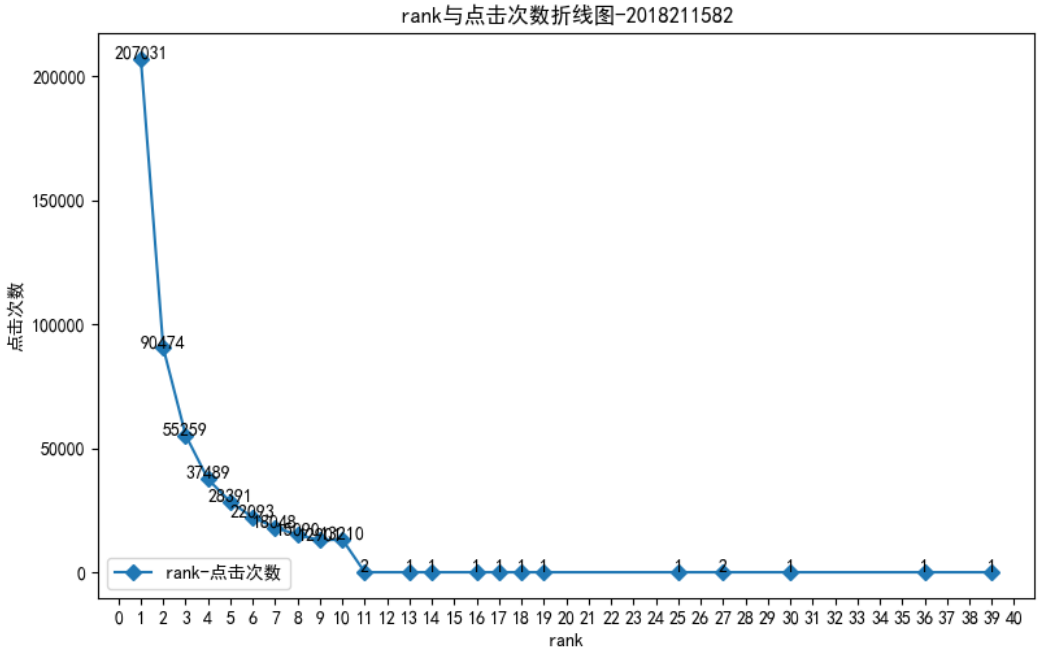
- **编写 Python 可视化代码：**
  根据结果，可以使用 PyHive 连接 hive 并将查询到的数据进行可视化展示，python 代码如下：

```
plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）
plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）
conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')
cursor = conn.cursor()
cursor.execute('select rank,count(*) as cnt from sogou_100w.sogou_ext_20111230 group by rank order by rank limit 30')
rank = []
clicks = []
for result in cursor.fetchall():
 rank.append(result[0])
 clicks.append(result[1])
cursor.close()
conn.close()
plt.plot(rank,clicks,marker='D')
plt.title('rank与点击次数折线图-2018211582')
plt.xlabel("rank")
plt.ylabel("点击次数")
plt.legend(['rank-点击次数'],loc=3)
x_major_locator=MultipleLocator(1)
#把x轴的刻度间隔设置为1，并存在变量里
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
for i,j in zip(rank,clicks):
    plt.text(i,j+2,"%d"%j,horizontalalignment='center')
plt.show()
```

- **结果分析：**

用户提交一个查询后，搜索引擎可能会返回多页结果，但是这些结果并不一定都是用户所想要的，因此用户一般不会将这些结果逐个都点击浏览。因此，根据搜索数据，我设计了 HiveSQL 语句，查询到了不同 rank 对应的点击次数，使用 python 远程连接 Hive 得到查询到数据并，并根据这个数据，制作了如下的折线图：





由折线图我们可以清晰的看出，99.9%的用户只会翻看搜索引擎返回结果的前 10 个结果，即返回结果页面的第一页，只有极少数的用户会翻看 rank10 之后的结果，仅有 14 次，翻看 rank 数最大为 39，仅有 1 次。这个

用户行为决定了尽管搜索引擎返回的结果数目十分庞大，但真正可能被绝大部门用户所浏览的，只有排在最前面的很小一部分而已，这也是一个 web 系统或网页需要做好 SEO 优化的原因，排名更靠前的返回结果更加受到使用者的青睐，并且这一概率很大。

2. 查询次数最多的前 20 个的用户 ID 和次数

- **设计挖掘目标**

  用户的查询次数一定程度上代表了用户对于一个搜索引擎产品的依赖程度和使用程度，同时也能够从侧面反映出搜索引擎的搜索算法的准确度，通过对于用户查询次数的分析，有利于优化搜索引擎，优化用户体验。因此，从庞大的数据中提取出查询次数最多的用户相关信息和次数是有必要做的一项。

- **完成 HiveSQL 代码**

```
SELECT uid,COUNT(*) AS cnt FROM sogou_ext_20111230 GROUP BY uid ORDER BY cnt D
ESC LIMIT 20;
```

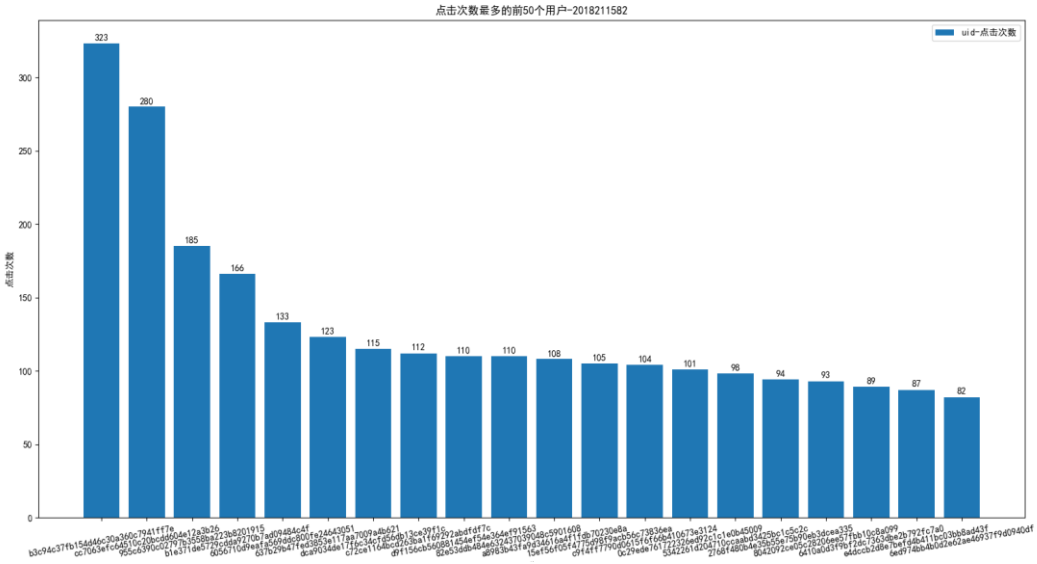
首先根据 uid 进行分组，统计出 uid 和每个 uid 的总共查询次数，再将根据查询次数降序排列，取前 20 个数据，得到查询次数最多的前 20 个用户的用户 ID

```
b3c94c37fb154d46c30a360c7941ff7e        323
cc7063efc64510c20bcdd604e12a3b26        280
955c6390c02797b3558ba223b8201915        185
b1e371de5729cdda9270b7ad09484c4f        166
6056710d9eafa569ddc800fe24643051        133
637b29b47fed3853e117aa7009a4b621        123
dca9034de17f6c34cfd56db13ce39f1c        115
c72ce1164bcd263ba1f69292abdfdf7c        112
d9f156cb560881454ef54e364ef91563        110
82e53ddb484e632437039048c5901608        110
a8983b43fa9d34616a4f1fdb70230e8a        108
15ef56f05f4775d98f9acb56c73836ea        105
c9f4ff7790d0615f6f66b410673e3124        104
0c29ede761722326ed92c1c1e0b45009        101
5342261d204710ccaabd3425bc1c5c2c        98
2768f480b4e35b55e75b90eb3dcea335        94
8042092ce05c28206ee57fbb10c8a099        93
6410a0d3f9bf2dc7363dbe2b792fc7a0        89
e4dccb2d8e7befd4b411bc03bb8ad43f        87
6ed974bb4b0d2e62ae46937f9d0940df        82
Time taken: 44.251 seconds, Fetched: 20 row(s)
```

- **编写 Python 可视化代码：**

  根据结果，可以编写 python 可视化代码如下，绘制柱状图

```
plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）
plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）
conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')
cursor = conn.cursor()
cursor.execute('SELECT uid,COUNT(*) AS cnt FROM sogou_100w.sogou_ext_20111230 GROUP BY uid ORDER BY cnt DESC LIMIT 20')
uid = []
clicks = []
for result in cursor.fetchall():
 uid.append(result[0])
 clicks.append(result[1])
cursor.close()
conn.close()
plt.bar(uid,clicks)
plt.title('点击次数最多的前50个用户-2018211582')
plt.xlabel("uid",fontsize=4)
plt.xticks(rotation = 10)
plt.ylabel("点击次数")
plt.legend(['uid-点击次数'],loc=1)
for i,j in zip(uid,clicks):
    plt.text(i,j+2,"%d"%j,horizontalalignment='center')
plt.show()
```



- **结果分析：**

由上述查询出的数据和柱状图展示，我们可以发现，查询总数前 20 的数字范围在 80 次以上，大概有 14 个用户查询次数在 100 次以上，查询次数最多的为用户 ID b3c94c37fb154d46c30a360c7941ff7e 所带来的 323 次，查询次数排名第 20 的总次数为 82 次，这些数据可以为公司提取到特殊的用户群体和用户画像，用来说明产品的使用率和用户对于产品的依赖性。

3. 直接搜索 url 的准确率分析

- **设计挖掘目标**

有时候，用户会直接输入 url 或 url 的一部分到检索框中进行检索，而此时能否对 url 进行特殊的识别也是考验搜索引擎的一个地方，因此我们可以通过对那些 keyword 中包含 url 的搜索数据进行分析，可以分析出目标：

用户点击的 URL 结果与用户输入的网址匹配的比例

- **完成 HiveSQL 代码**

```
SELECT SUM(IF(instr(url,keyword)<=0,1,0)),SUM(IF(instr(url,keyword)>0,1,0)) F
ROM (SELECT * FROM sogou_100w.sogou_ext_20111230 WHERE keyword LIKE '%www%')
a;
```

首先，使用 LIKE 语句筛选出 keyword 中包含'www'的数据，这些都是用户直接输入网址进行查询的数据，之后我们使用 instr 子串检测函数，检测 url 中是否包含 keyword，使用 TF 语句：

对于第一项，我们要统计不包含的数据项个数，因此若子串结果为负，则返回 1，否则返回 0，以此统计 url 中不含有关键词的数据个数。

对于第二项，统计包含的数据项个数，如果子串检测结果为正，则返回 1，否则返回 0，以此统计 url 含有关键词的数据个数。

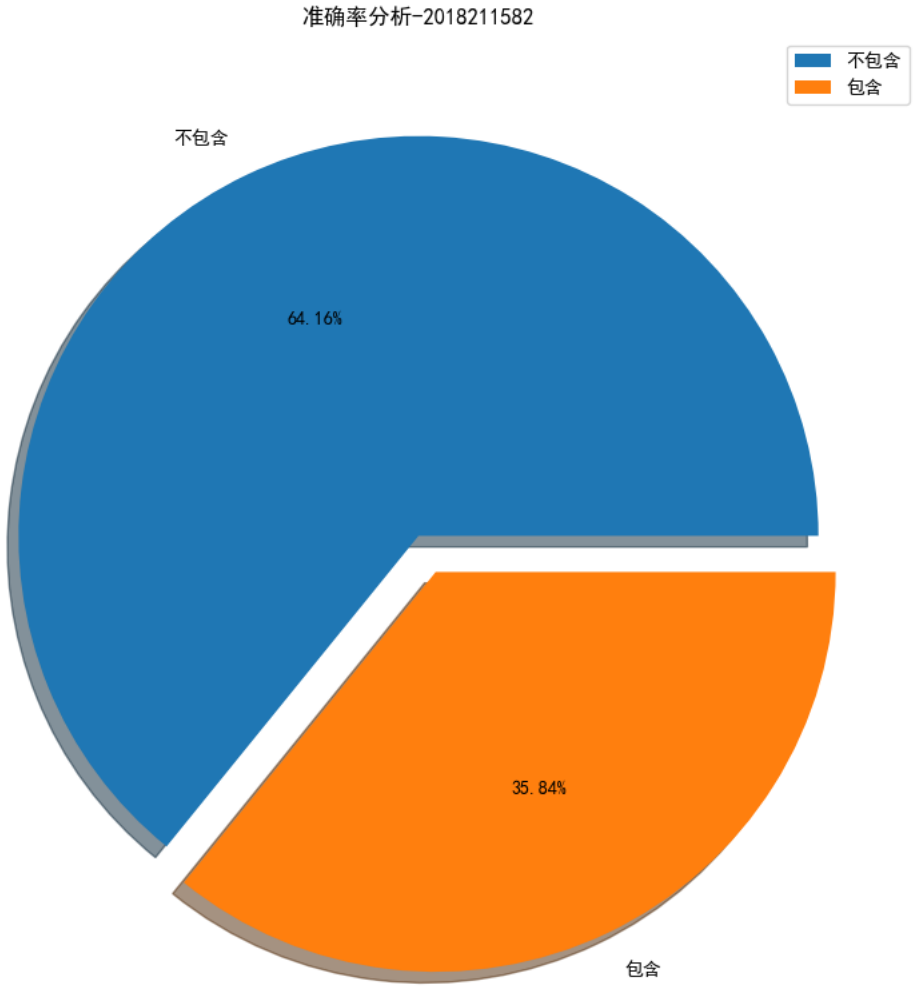
使用两个 SUM 函数统计总数并返回。

```
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.49 sec   HDFS Read: 64432719 HDFS Write: 109 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 490 msec
OK
4881    2727
Time taken: 17.266 seconds, Fetched: 1 row(s)
```

- **编写 Python 可视化代码：**

根据结果，可以编写 python 可视化代码如下，绘制饼图

```
plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）
plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）
conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')
cursor = conn.cursor()
cursor.execute("SELECT SUM(IF(instr(url,keyword)<=0,1,0)),SUM(IF(instr(url,keyword)>0,1,0)) FROM (SELECT * FROM sogou_100w.
sum = []
for result in cursor.fetchall():
 sum.append(result[0])
 sum.append(result[1])
cursor.close()
conn.close()
labels=['不包含','包含']
exp = [0, 0.1]
plt.pie(x=sum,labels=labels,explode=exp,autopct="%0.2f%%",shadow=True)
plt.legend()
plt.title('准确率分析-2018211582')
plt.show()
```



- **结果分析：**

由上述饼状图和查询数据可以看出，用户查询关键词为网址时，用户最后点击 URL 中包含该 keyword 的数据项共有 2727 项，占比 35.84%，用户最后点击的 URL 中不包含该 keyword 的数据项共有 4881 项，占比 64.16%。由此可以看出，若用户直接输入网址，以此为关键词查询，搜索出的结果似乎并不能够满足用户的需要，用户想搜索的网址很多并没有被检索到，这一方面可能是搜素引擎搜索算法的原因，另一方面也间接的表明，用户通过直接输入网址查询的方式并不理想，效果没有输入一些检索词而得到的准确。

4. 用户使用时间挖掘

- **设计挖掘目标**

通过对用户使用时间的获取，将使用时间范围分为早上、中午、晚上和午夜，分别代表 8-12 点、12-18 点、18-23 点、0 点-8 点。通过对使用时间的分析，可以挖掘出用户使用频率最高的时间段

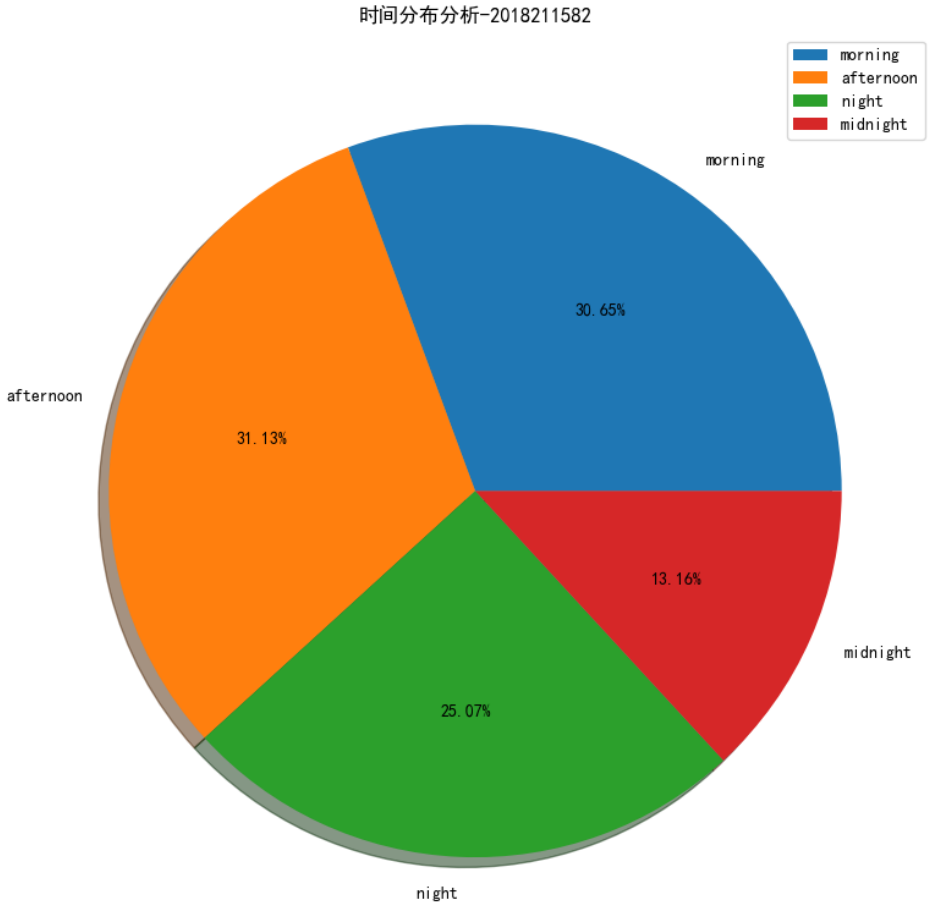
- 完成 HiveSQL 代码

```
SELECT HOUR FROM sogou_100w.sogou_ext_20111230
```

直接返回 huor 结果，之后使用 python 进行时间划分并展示。

- 编写 Python 可视化代码：
  根据结果，可以编写 python 可视化代码如下，绘制饼图

```
conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')
cursor = conn.cursor()
cursor.execute("select hour from sogou_100w.sogou_ext_20111230")
hours = []
morning = []
midnight =[]
afternoon = []
night = []
for result in cursor.fetchall():
    hours.append(result[0])

for i in range(len(hours)):
    if(int(hours[i])>8 and int(hours[i])<=12):
        morning.append(int(hours[i]))
    if(int(hours[i])>12 and int(hours[i]) <=18):
        afternoon.append(int(hours[i]))
    if(int(hours[i])>18 and int(hours[i]) <=23):
        night.append(int(hours[i]))
    if(int(hours[i])>0 and int(hours[i]) <= 8):
        midnight.append(int(hours[i]))
    #hours.append(times[i][8:10])

x=[]
x.append(len(morning))
x.append(len(afternoon))
x.append(len(night))
x.append(len(midnight))
cursor.close()
conn.close()
labels=['morning','afternoon','night','midnight']
plt.pie(x,labels=labels,autopct="%0.2f%%",shadow=True)
plt.title('时间分布分析-2018211582')
plt.legend(loc='upper right')
plt.show()
```



- **结果分析：**

可以看到30.65%的用户会选择在早上使用搜索引擎，31.13%的用户选择在下午只用搜索引擎，25.07%的用户选择在晚上使用搜索引擎，13.16%的用户选择在午夜使用搜索引擎。整体来看，用户偏向早上和下午即白天时间使用，午夜使用的用户较少。

附　　录

实验分析 python 代码：

一、分析 rank 与用户点击次数之间的关系

```
from matplotlib import lines

from pyhive import hive

import matplotlib.pyplot as plt

from matplotlib.pyplot import MultipleLocator

plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）

plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）

conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')

cursor = conn.cursor()

cursor.execute('select rank,count(*) as cnt from sogou_100w.sogou_ext_20111230 group by rank order by rank limit 30')

rank = []

clicks = []

for result in cursor.fetchall():

 rank.append(result[0])

 clicks.append(result[1])

cursor.close()

conn.close()

plt.plot(rank,clicks,marker='D')

plt.title('rank 与点击次数折线图-2018211582')

plt.xlabel("rank")
```

```
plt.ylabel("点击次数")

plt.legend(['rank-点击次数'],loc=3)

x_major_locator=MultipleLocator(1)

#把 x 轴的刻度间隔设置为 1，并存在变量里

ax=plt.gca()

ax.xaxis.set_major_locator(x_major_locator)

for i,j in zip(rank,clicks):

    plt.text(i,j+2,"%d"%j,horizontalalignment='center')

plt.show()
```

二、查询次数最多的前 20 个的用户 ID 和次数

```
from matplotlib import lines

from pyhive import hive

import matplotlib.pyplot as plt

from matplotlib.pyplot import MultipleLocator

plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）

plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）

conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')

cursor = conn.cursor()

cursor.execute('SELECT uid,COUNT(*) AS cnt FROM sogou_100w.sogou_ext_20111230 GROUP BY uid ORDER BY cnt DESC LIMIT 20')

uid = []

clicks = []
```

```
for result in cursor.fetchall():

 uid.append(result[0])

 clicks.append(result[1])

cursor.close()

conn.close()

plt.bar(uid,clicks)

plt.title('点击次数最多的前 50 个用户-2018211582')

plt.xlabel("uid",fontsize=4)

plt.xticks(rotation = 10)

plt.ylabel("点击次数")

plt.legend(['uid-点击次数'],loc=1)

for i,j in zip(uid,clicks):

    plt.text(i,j+2,"%d"%j,horizontalalignment='center')

plt.show()
```

三. 直接搜索 url 的准确率分析

```
from matplotlib import lines

from pyhive import hive

import matplotlib.pyplot as plt

from matplotlib.pyplot import MultipleLocator

plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）

plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）
```

```
conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')

cursor = conn.cursor()

cursor.execute("SELECT SUM(IF(instr(url,keyword)<=0,1,0)),SUM(IF(instr(url,keyword)>0,1,0)) FROM (SELECT * FROM sogou_100w.sogou_ext_20111230 WHERE keyword LIKE '%www%') a")

sum = []

for result in cursor.fetchall():

 sum.append(result[0])

 sum.append(result[1])

cursor.close()

conn.close()

labels=['不包含','包含']

exp = [0, 0.1]

plt.pie(x=sum,labels=labels,explode=exp,autopct="%0.2f%%",shadow=True)

plt.legend()

plt.title('准确率分析-2018211582')

plt.show()
```

四、用户使用时间点分布

```
from matplotlib import lines

from pyhive import hive

import matplotlib.pyplot as plt

from matplotlib.pyplot import MultipleLocator

plt.rcParams['font.sans-serif'] = ['SimHei'] # 步骤一（替换 sans-serif 字体）
```

```
plt.rcParams['axes.unicode_minus'] = False # 步骤二（解决坐标轴负数的负号显示问题）

conn = hive.Connection(host='119.3.212.133',port=10000,auth='NOSASL',username='root')

cursor = conn.cursor()

cursor.execute("select hour from sogou_100w.sogou_ext_20111230")

hours = []

morning = []

midnight =[]

afternoon = []

night = []

for result in cursor.fetchall():

    hours.append(result[0])



for i in range(len(hours)):

    if(int(hours[i])>8 and int(hours[i])<=12):

        morning.append(int(hours[i]))

    if(int(hours[i])>12 and int(hours[i]) <=18):

        afternoon.append(int(hours[i]))

    if(int(hours[i])>18 and int(hours[i]) <=23):

        night.append(int(hours[i]))

    if(int(hours[i])>0 and int(hours[i]) <= 8):

        midnight.append(int(hours[i]))

    #hours.append(times[i][8:10])



x=[]
```

```
x.append(len(morning))

x.append(len(afternoon))

x.append(len(night))

x.append(len(midnight))

cursor.close()

conn.close()

labels=['morning','afternoon','night','midnight']

plt.pie(x,labels=labels,autopct="%0.2f%%",shadow=True)

plt.title('时间分布分析-2018211582')

plt.legend(loc='upper right')

plt.show()
```