小组：嘤（yíng）嘤嘤

2018.8

大数据分析项目报告书

一、项目简介

1、项目概述 4

2.基本构架 4

二、基于Hadoop、Hive的游戏日志分析

1、项目基本介绍 5

2、需求详解 5

3、结果分析 9

三、基于Hadoop、Hive的玩家游戏表现分析

1、项目基本介绍 17

2、需求详解 18

3、结果分析 20

四、基于Hadoop、Hive的电商交易日志分析

1、项目基本介绍 43

2、需求详解 44

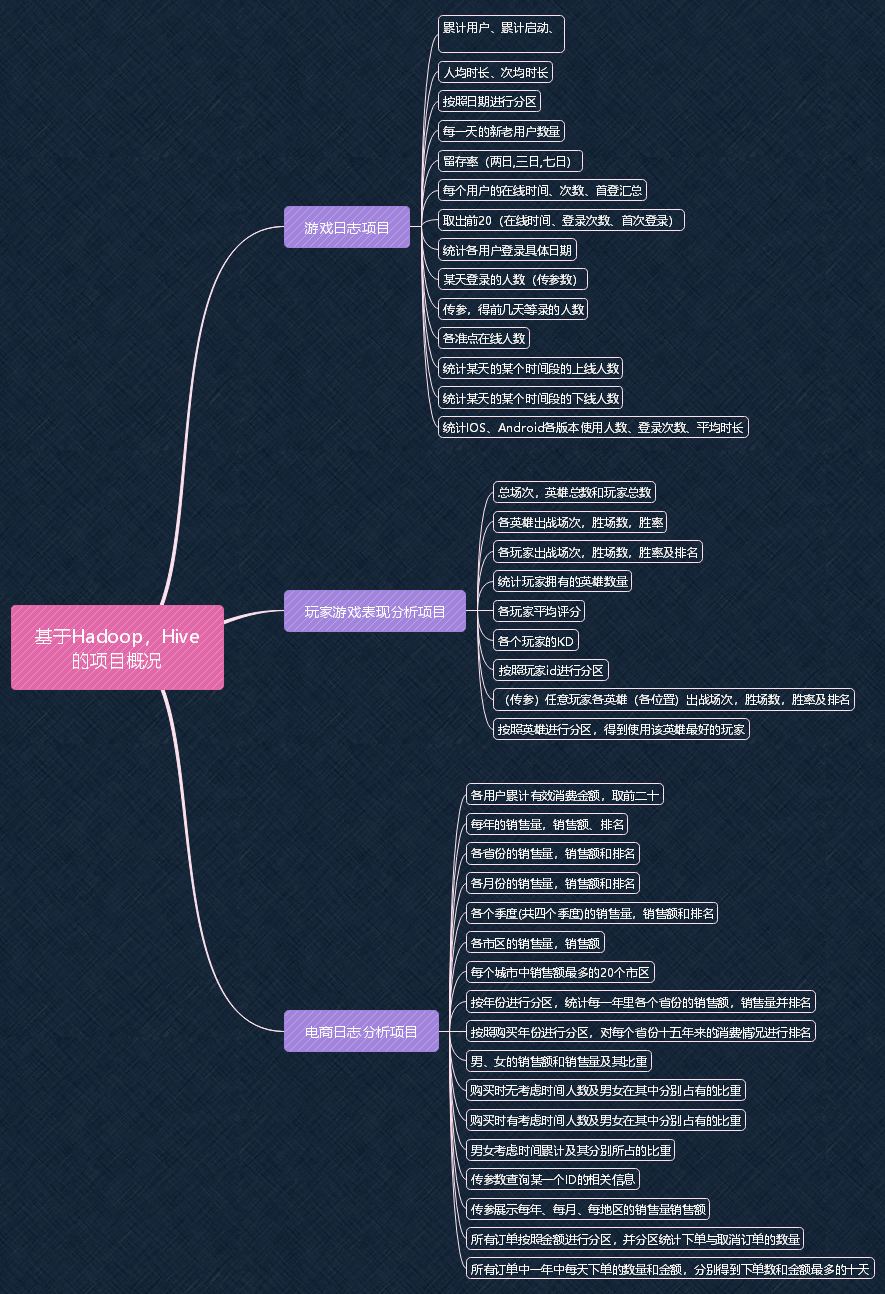
3、结果分析 58

一、项目简介

1、项目概述

本报告主要针对基于Hadoop、Hive的游戏日志分析、玩家游戏表现分析、电商交易日志分析三个项目进行阐述，旨在通过对数据提出需求并进行分析，用直观的形式表现出来最终得出相应的分析。

1. 基本构架



二、基于Hadoop、Hive的游戏日志分析

1、基本项目介绍

原始数据格式为



已生成极低可信度的说明

分别代表的含义：

0-用户id、1-操作系统、2-系统编号、3-上线时间、4-下线时间、5-累计在线时长

2、需求详解

（1）TotalInfo.java累计用户、累计启动

mapreduce代码思路：累计用户为用户唯一ID不重复出现的次数，在map端聚合之后计算所有的reduce次数即可。累计启动在每个reduce的迭代器里累加1即可求得。

1. AverageInfo.java人均时长、次均时长

mapreduce代码思路：id聚合后在每个迭代器里累加每次记录的在线时长。最后除以人数和启动次数即可。

1. DataPartion.java按照日期进行分区，每日登陆总数

mapreduce代码思路：通过日期觉得Partition的返回值。从而实现按日期分区。

1. EverydayNewAndOld.java每一天的新老用户数量。

mapreduce代码思路：按照id聚合后，循环判断出用户第一次登陆的日期，则用户在那一天为新用户，相应的日期数组加一。

1. ActiveRate.java留存率（两日,三日,七日）

mapreduce代码思路：例如三日留存率：按照id聚合后，判断用户是否在那三天有过登陆记录，有则总用户加一。若该用户在那三天都有登陆记录则count加一，最后做除法即可。

1. Top20Info.java 每个用户的在线时间、次数、首登汇总

mapreduce代码思路：通过id聚合。循环累加用户的在线时间，在迭代器里取第一次的登陆时间为首登时间。循环的次数为上线次数。

1. Top20.java取出前20（在线时间、登录次数、首次登录）

mapreduce代码思路：通过把上述信息连成的字符串作为key值，Sort函数中通过split取出相关信息并比较，决定返回值，从而实现排序。

1. Authority.java统计各用户在哪天登陆了

mapreduce代码思路：通过id聚合，循环判读该用户登陆的天数，并付给相应的变量false/true值。循环之后将true对应的天数输出。

1. OnlineDay.java某天登录的人数（传参数）

mapreduce代码思路：通过上述文件，在ruduce中判断该天是否有登录，决定是否累加登录人数。

(10 )BeforeX.java传参，得前几天登录的人数

mapreduce代码思路：id聚合后，通过传入的参数判断该用户该日期之前是否有登录。从而累加相应变量。

1. OnlineHour.java七天各准点在线人数

mapreduce代码思路：对初始数据进行处理。通过日期以及时间对上线时间和下线时间的跨度进行判断。得出它在哪些连续的整点有在线记录。从而将相关数组进行累加。

（12）LogInCount.java统计某天的某个时间段的上线人数，时间段分为

上午(6-12)，下午(12-18)，晚(18-24)，凌晨(0-6)

四个时间段，求出各时间段的比重

mapreduce代码思路：通过读取相应日期的文件，通过id聚合，判断该用户在四个时段中哪个时段有上线记录，从而将相应数组加一。

（13）LogOffCount.java统计某天的某个时间段的下线人数，时间段分为

上午(6-12)，下午(12-18)，晚(18-24)，凌晨(0-6)

四个时间段，求出各时间段的比重

mapreduce代码思路：通过读取相应日期的文件，通过id聚合，判断该用户在四个时段中哪个时段有下线记录，从而将相应数组加一。

1. OSCount.jave统计IOS、Android使用人数,各版本登录次数和平均时长

mapreduce代码思路：将系统作为key值，在reduce中迭代即可求的两大系统的人数。通过将系统和版本号的连接成的字串作为key值，即可在reduce中求的各版本的累计时间和登陆次数。

hive代码实现过程

1.累计用户

select COUNT(DISTINCT user\_id) from user\_test2; 33266

2。累计启动次数

select COUNT(user\_id) from user\_test2;333886

3.每个用户的id、在线时间、次数 ffff5256-5b6e-44f9-a33c-09d10f5cb81c 248560 11

drop table if exists id\_test1;

create table if not exists id\_test1 as

select user\_id as id,SUM(cast(time as int)) as time,count(user\_id) as counts from user\_test2

group by user\_id;

4.每个用户（在线时间、登录次数、）前20 排序

select \* from id\_test1 order by time desc,counts desc limit 20;

6.每天登录的人数

select COUNT(DISTINCT user\_id) from user\_test2 where logout like '2017-01-01T%';

7.前三天登录的人数

select COUNT(DISTINCT user\_id)from user\_test2 where logout like '2017-01-01T%' or logout like '2017-01-02T%' or logout like '2017-01-03T%';

8.系统比率（iOS，Android）

drop table if exists system\_test;

create table if not exists system\_test as

select system as system,count(1) as total from user\_test2

group as system;

//rate: 系统总数／总次数（利用前面的结果）

3、结果分析  
（1）七日用户数据

|  |  |  |  |
| --- | --- | --- | --- |
| 累计用户 | 累计登陆 | 人均时长 | 次均时长 |
| 33266 | **333886** | **245605** | **24470** |

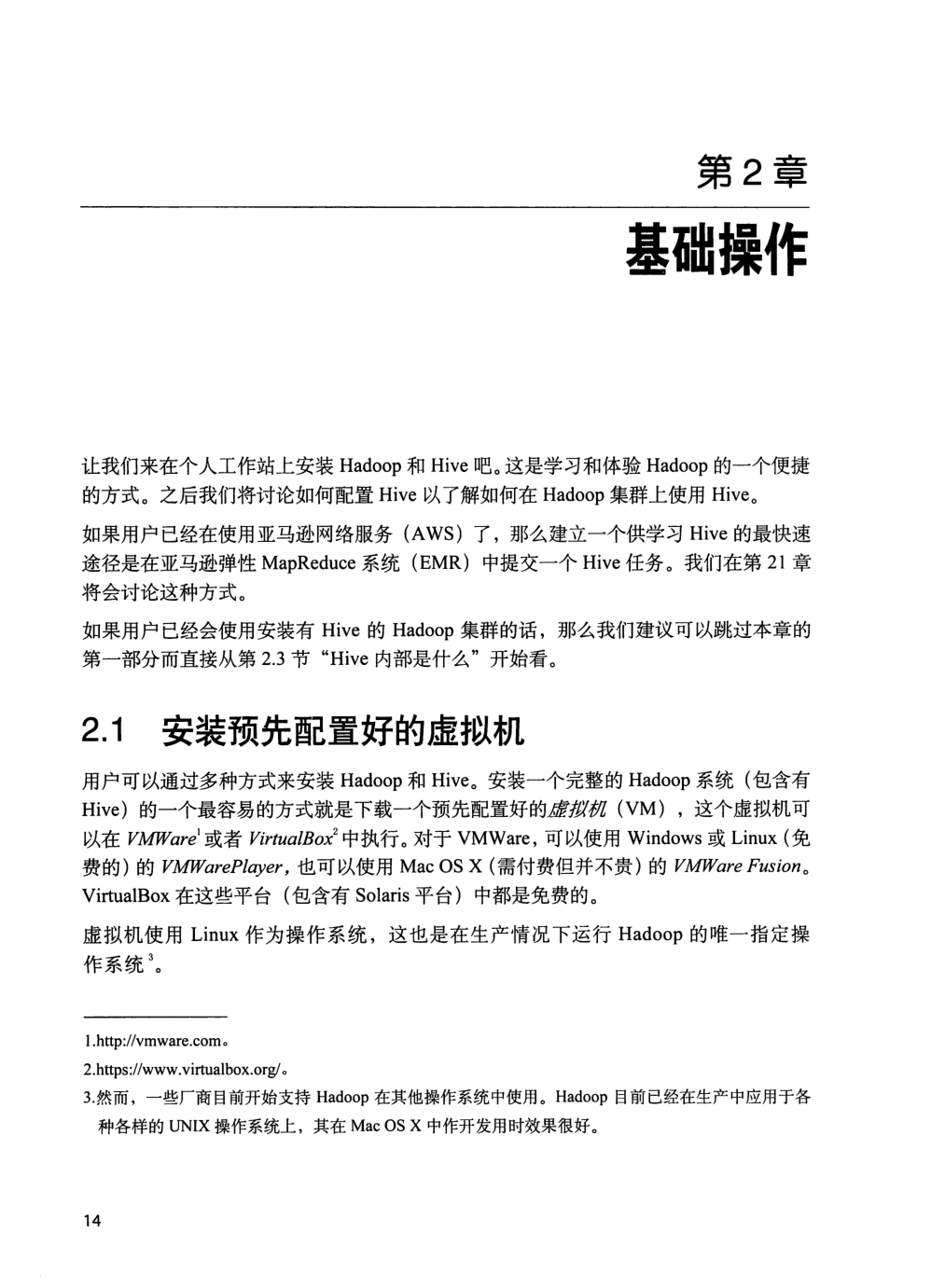
七日用户总数据可以宏观地看出游戏吸引玩家的总体情况，人均时长和次均时长是反映用户游戏体验的重要指标，时长越长，代表游戏越有吸引力，可以根据每7天登陆的用户和人均/次均时长变化趋势进行游戏完善分析。

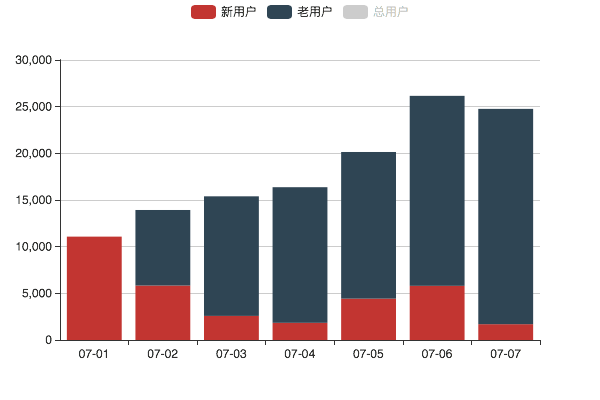
(2)按照日期分区后统计每日用户

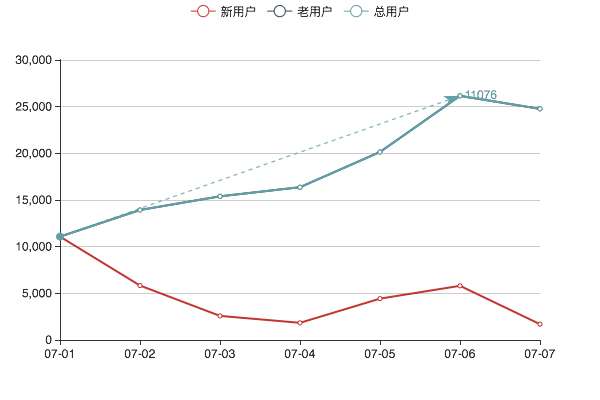
（与需求3一起展示）

（3）每日的新老用户及总用户

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 01-01 | 01-02 | 01-03 | 01-04 | 01-05 | 01-06 | 01-07 |
| 新用户 | 11076 | 5835 | 2582 | 1841 | 4432 | 5804 | 1696 |
| 老用户 | 0 | 8095 | 12812 | 14530 | 15712 | 20355 | 23067 |
| 总用户 | 11076 | 13930 | 15394 | 16371 | 20144 | 26159 | 24763 |







由新老用户占比和七日的变化趋势可以看出，游戏的总用户数整体呈上升趋势，新用户数量存在波动，老用户的数量在持续增加，说明游戏存在吸引玩家的亮点，继续开发将持续吸引用户，进而提高用户的留存率。

1. 留存率

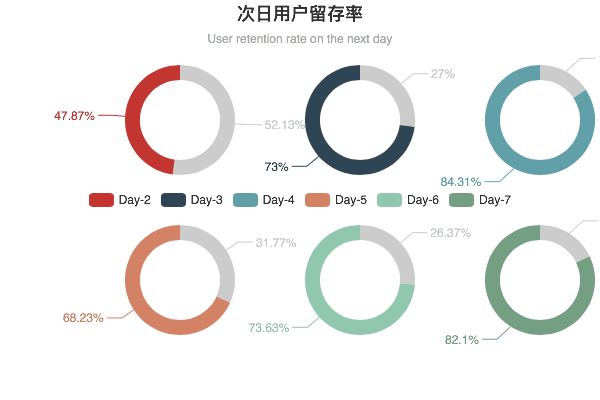
次日留存率：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Day-2 | Day-3 | Day-4 | Day-5 | Day-6 | Day-7 |
| 47.87% | 73.00% | 84.31% | 68.23% | 73.63% | 82.10% |

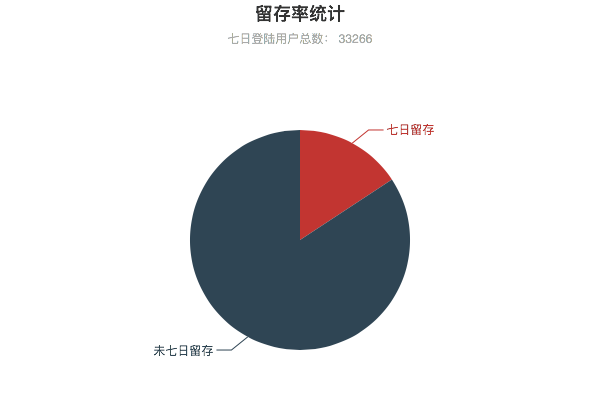
三日留存率：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Day-3 | Day-4 | Day-5 | Day-6 | Day-7 |
| 36.83% | 62.17% | 58.77% | 51.60% | 60.61% |

七日留存率：15.74%





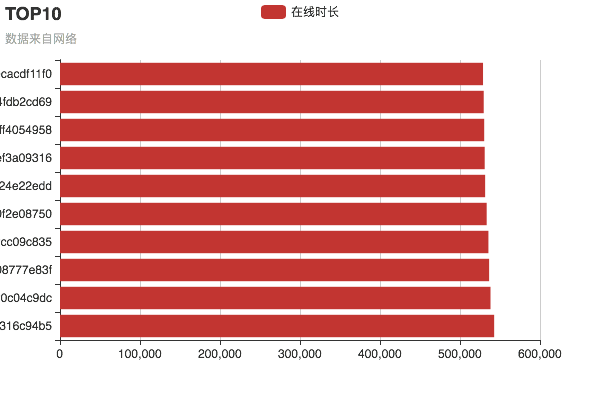


通过观察两日、三日和七日用户的留存率，可以得出游戏对玩家的吸引程度。两日和三日留存率基本在50%以上，说明有接近一半的玩家会连续两天或三天登陆游戏，态势较好，且留存率整体有上升趋势，但七日留存率有待提高，应该开发更多的亮点，比如每日签到发放系统奖励。

1. 按照时间进行排名并取出TOP10

|  |  |  |  |
| --- | --- | --- | --- |
| 用户ID | 登陆时长 | 次数 | 首登时间 |
| 1c0f9309-390c-40ab-9aac-69dc316c94b5 | 542695 | 21 | 2017-01-04T22:13:44 |
| 4baeaff5-2df4-4650-ad03-59b20c04c9dc | 538126 | 24 | 2017-01-01T23:59:52 |
| 8d01e974-9da4-4f84-a864-05d08777e83f | 536395 | 30 | 2017-01-02T03:17:57 |
| 17cc669e-a347-40c2-a725-cd32cc09c835 | 535517 | 21 | 2017-01-03T00:21:41 |
| e4a5b102-8845-4e2b-b511-d5b0f2e08750 | 533358 | 21 | 2017-01-03T23:59:56 |
| c80498c5-c2c7-42f6-8cf3-863424e22edd | 531422 | 15 | 2017-01-05T20:09:32 |
| 53f04a86-0dcd-4afe-8729-96cef3a09316 | 530883 | 12 | 2017-01-03T00:04:26 |
| 757fe9b1-11ea-4d51-a24c-1d8ff4054958 | 530242 | 13 | ]2017-01-06T00:04:52 |
| cc372845-9be3-4626-b887-5ee4fdb2cd69 | 529654 | 15 | 2017-01-07T00:55:15 |
| 456fa78a-aa9a-4240-b8cf-4becacdf11f0 | 528784 | 14 | 2017-01-06T07:49:02 |
| 818ed4d9-749c-4ea1-82c0-ce3b656fb951 | 528353 | 11 | 2017-01-03T01:51:49 |
| c7bddf00-201b-42f5-af67-caa81ae9da46 | 527156 | 19 | 2017-01-05T23:57:44 |
| 31a05c82-22f8-4d3d-9c0e-0e378c342e09 | 526387 | 17 | 2017-01-07T23:03:43 |
| a5a94157-723c-4518-997e-6d2353faa13c | 526167 | 12 | 2017-01-04T23:23:41 |
| 555b28d8-766f-4b32-92cd-85a9652d87be | 525633 | 10 | 2017-01-01T00:34:51 |
| 0f6b52d9-8344-4269-9331-650a773cc172 | 525392 | 21 | 2017-01-04T01:18:06 |
| 55f8b325-52c4-478e-9f0c-555eda0edc8d | 524241 | 13 | 2017-01-05T11:55:24 |
| dec163f1-93ba-47d8-a1a7-d35c270d9c76 | 522063 | 22 | 2017-01-01T06:30 |
| 528a4f14-6b87-4d96-a2dc-c0a6e3a7bd19 | 521953 | 10 | 2017-01-06T21:10:51 |
| 4c8d3baf-bbdb-41fb-9c6c-318bba9874f7 | 521785 | 15 | 2017-01-03T15:08:04 |

得出TOP20的信息后，筛选出TOP10进行展示，由于时长按照由低到高排列，不存在时长相同需要比较次数或首登时间的情况，所以以下柱状图按照时长排列，由下到上依次为第一名到第十名

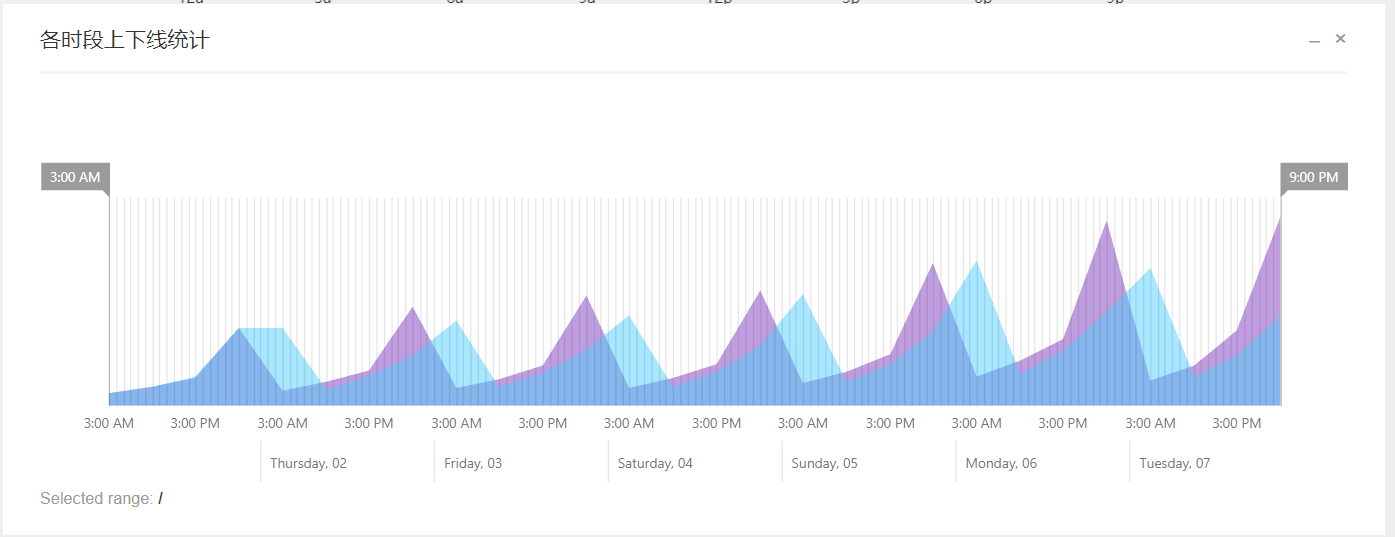


1. 七日24个整点在线人数

用mapreduce算出每日每24个整点的在线人数，并展示在散点图上，可以直观看出每日玩家登陆时间集中在中间时段，也可以看出七日的总用户数呈上升趋

纵向观察，1号到7号的散点越来越密集，表示七天内用户数越来越多，游戏的宣传与完善达到了一定的效果；横向观察，可以看出每日用户在线时间集中在晚上到凌晨的时间段，结合下图。

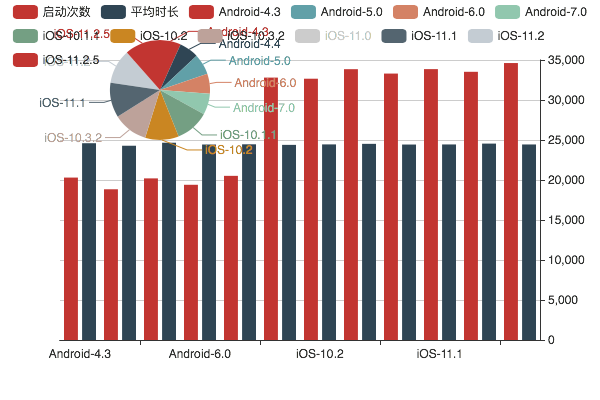
1. 四个时间段的上线和下线人数

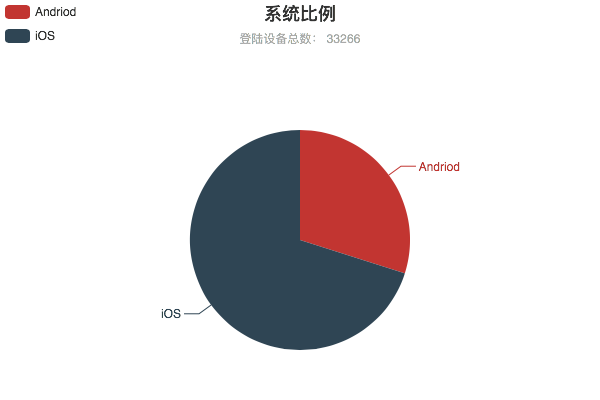


各时间段上下线用户集中区域有一定的规律，上线时间主要集中在晚上六点左右，而下线时间主要集中在凌晨三点左右，可以在这个时间段内多发放系统福利，创新游戏模式来吸引玩家，针对下线时间过晚，也可以设置未成年人防沉迷模式。

1. Android和iOS系统及版本用户使用比例

设备总数：33266 其中Andriod ：9947 iOS ：23319





分析可知，不同系统及版本的用户数存在差异，其中iOS用户显著多于Andriod用户，但各系统平均登陆时长差异不大，说明游戏的兼容性良好，不存在闪退等问题。针对Android用户相对较少，可以对Andriod版本进行一定的优化，并在Andriod软件商店加大游戏的宣传力度，吸引更多的Andriod用户。

1. 基于Hadoop、Hive的玩家游戏表现分析

1、基本项目介绍

（1）背景分析：MOBA（Multiplayer Online Battle Arena）类游戏是一种以竞技场为游戏场景的多人在线即时游戏`游戏通常以5v5的对抗形式展开，每个玩家可以操控自己的游戏角色来与对方抗衡、争夺地图资源。摧毁敌方水晶判定为游戏获胜。

MOBA类游戏有公平竞争和即时对抗的特性，被许多玩家追捧。MOBA类游戏现在向着游戏资源（金钱）更容易获得、游戏内事业更简单、英雄及技能更简单的方向发展。MOBA类游戏中目前玩家最多的时英雄联盟（LOL），本项目就是以英雄联盟5v5经典模式为数据基础进行处理和分析的。

（2）原始数据含义及解释

Dr\_Mundo 英雄名（英雄唯一标识）

0 胜负（胜为1）

88 玩家编号（1-100）

Help 位置（Top/Ranger/Mid/Down/Help）

Tank 英雄职业（Fighter/AP/ADC/Support/Tank）

5 杀敌数（kill）

18 死亡数（die）

6 评分（grade）

1. 房间号（room）

2、需求详解

（1）求出总场次，英雄总数和玩家总数。

AllCount.java

（2）各英雄出战场次，胜场数，胜率。排名规则：胜率>胜场数>玩家编号

AllHeroInfo.java & HeroSort.java

（3）各玩家出战场次，胜场数，胜率及排名。

AllPlayerInfo.java & PlayerSort.java

（4）统计玩家拥有的英雄数量

PlayerHeroCount.java

（5）各玩家平均评分

PlayerGrade.java

（6）各个玩家的KD

PlayerKD.java

（7）按照玩家id进行分区

PlayerPartition.java

（8）传参（玩家id)，求出任意玩家各英雄（各位置）出战场次，胜场数，胜率及排名

SearchPlayerInfo.java SearchPlayerPosition.java

（9）按照英雄进行分区，得到使用该英雄最好的玩家

PartitionHero.java GreatPlayer.java

实现方法及代码

（1）在reduce端使用HashSet对文件中局数、玩家数和英雄数进行累计。

（2）Map端以hero为key进入reduce进行数据整合，以AllHeroInfo的输出为输入文件，对胜率排序取前十。

（3）对AllHeroInfo的输出文件进行操作，同（2），得到PlayerTop10

（4）Map端以玩家编号为key输出进入reduce，使用HashSet保证英雄无重复

（5）Map端以玩家编号为key输出进入reduce，对该玩家每局评分累加求平均

（6）Map端以玩家编号为key输出进入reduce，累计玩家的kill & death，得到KD比

（7）玩家编号为1-100，与分区编号0-99相对应

（8）在map端进行cleanup操作，在主函数中键入编号，查询结果

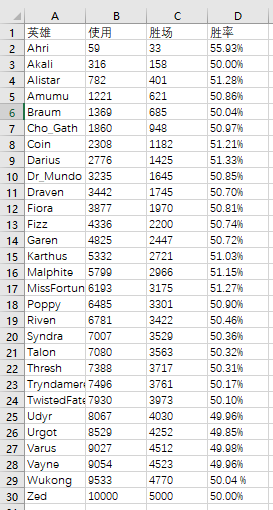
（9）对英雄分区后的数据进行以胜率进行排序

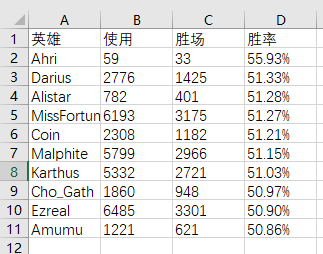
1. 结果分析

注：与模块2对应



（2）全部英雄使用场次/胜场/胜率 + 胜率Top10





（3）各玩家出战场次，胜场数，胜率及排名

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 编号 | 总场次 | 胜场 | 胜率 |
| 1 | 0 | 113 | 52 | 46.02% |
| 2 | 1 | 85 | 41 | 48.24% |
| 3 | 10 | 113 | 55 | 48.67% |
| 4 | 11 | 103 | 52 | 50.49% |
| 5 | 12 | 94 | 46 | 48.94% |
| 6 | 13 | 97 | 55 | 56.70% |
| 7 | 14 | 107 | 58 | 54.21% |
| 8 | 15 | 96 | 50 | 52.08% |
| 9 | 16 | 103 | 52 | 50.49% |
| 10 | 17 | 96 | 58 | 60.42% |
| 11 | 18 | 106 | 53 | 50.00% |
| 12 | 19 | 87 | 43 | 49.43% |
| 13 | 2 | 102 | 52 | 50.98% |
| 14 | 20 | 110 | 56 | 50.91% |
| 15 | 21 | 74 | 30 | 40.54% |
| 16 | 22 | 110 | 54 | 49.09% |
| 17 | 23 | 105 | 53 | 50.48% |
| 18 | 24 | 96 | 44 | 45.83% |
| 19 | 25 | 110 | 60 | 54.55% |
| 20 | 26 | 102 | 46 | 45.10% |
| 21 | 27 | 112 | 62 | 55.36% |
| 22 | 28 | 92 | 38 | 41.30% |
| 23 | 29 | 102 | 46 | 45.10% |
| 24 | 3 | 99 | 46 | 46.46% |
| 25 | 30 | 95 | 45 | 47.37% |
| 26 | 31 | 108 | 51 | 47.22% |
| 27 | 32 | 106 | 56 | 52.83% |
| 28 | 33 | 98 | 43 | 43.88% |
| 29 | 34 | 107 | 61 | 57.01% |
| 30 | 35 | 93 | 47 | 50.54% |
| 31 | 36 | 97 | 54 | 55.67% |
| 32 | 37 | 101 | 50 | 49.50% |
| 33 | 38 | 92 | 42 | 45.65% |
| 34 | 39 | 111 | 47 | 42.34% |
| 35 | 4 | 86 | 47 | 54.65% |
| 36 | 40 | 112 | 56 | 50.00% |
| 37 | 41 | 82 | 41 | 50.00% |
| 38 | 42 | 104 | 45 | 43.27% |
| 39 | 43 | 84 | 42 | 50.00% |
| 40 | 44 | 94 | 43 | 45.74% |
| 41 | 45 | 96 | 49 | 51.04% |
| 42 | 46 | 105 | 47 | 44.76% |
| 43 | 47 | 106 | 50 | 47.17% |
| 44 | 48 | 117 | 62 | 52.99% |
| 45 | 49 | 92 | 57 | 61.96% |
| 46 | 5 | 123 | 68 | 55.28% |
| 47 | 50 | 111 | 73 | 65.77% |
| 48 | 51 | 102 | 53 | 51.96% |
| 49 | 52 | 89 | 45 | 50.56% |
| 50 | 53 | 115 | 68 | 59.13% |
| 51 | 54 | 86 | 40 | 46.51% |
| 52 | 55 | 115 | 60 | 52.17% |
| 53 | 56 | 102 | 58 | 56.86% |
| 54 | 57 | 93 | 38 | 40.86% |
| 55 | 58 | 79 | 37 | 46.84% |
| 56 | 59 | 86 | 48 | 55.81% |
| 57 | 6 | 102 | 54 | 52.94% |
| 58 | 60 | 105 | 52 | 49.52% |
| 59 | 61 | 95 | 39 | 41.05% |
| 60 | 62 | 106 | 45 | 42.45% |
| 61 | 63 | 88 | 41 | 46.59% |
| 62 | 64 | 95 | 41 | 43.16% |
| 63 | 65 | 120 | 48 | 40.00% |
| 64 | 66 | 99 | 45 | 45.45% |
| 65 | 67 | 110 | 68 | 61.82% |
| 66 | 68 | 89 | 41 | 46.07% |
| 67 | 69 | 107 | 46 | 42.99% |
| 68 | 7 | 106 | 59 | 55.66% |
| 69 | 70 | 109 | 44 | 40.37% |
| 70 | 71 | 97 | 56 | 57.73% |
| 71 | 72 | 84 | 44 | 52.38% |
| 72 | 73 | 95 | 49 | 51.58% |
| 73 | 74 | 90 | 44 | 48.89% |
| 74 | 75 | 103 | 46 | 44.66% |
| 75 | 76 | 95 | 50 | 52.63% |
| 76 | 77 | 102 | 58 | 56.86% |
| 77 | 78 | 96 | 55 | 57.29% |
| 78 | 79 | 106 | 52 | 49.06% |
| 79 | 8 | 106 | 54 | 50.94% |
| 80 | 80 | 116 | 71 | 61.21% |
| 81 | 81 | 94 | 53 | 56.38% |
| 82 | 82 | 110 | 54 | 49.09% |
| 83 | 83 | 99 | 57 | 57.58% |
| 84 | 84 | 103 | 43 | 41.75% |
| 85 | 85 | 101 | 50 | 49.50% |
| 86 | 86 | 93 | 42 | 45.16% |
| 87 | 87 | 97 | 42 | 43.30% |
| 88 | 88 | 112 | 61 | 54.46% |
| 89 | 89 | 95 | 50 | 52.63% |
| 90 | 9 | 89 | 43 | 48.31% |
| 91 | 90 | 88 | 41 | 46.59% |
| 92 | 91 | 102 | 51 | 50.00% |
| 93 | 92 | 101 | 44 | 43.56% |
| 94 | 93 | 85 | 41 | 48.24% |
| 95 | 94 | 103 | 50 | 48.54% |
| 96 | 95 | 109 | 53 | 48.62% |
| 97 | 96 | 99 | 46 | 46.46% |
| 98 | 97 | 88 | 45 | 51.14% |
| 99 | 98 | 111 | 58 | 52.25% |
| 100 | 99 | 99 | 49 | 49.49% |

（4）统计玩家拥有英雄的数量

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 英雄数量 | 编号 | 英雄数量 |
| 0 | 15 | 54 | 20 |
| 1 | 21 | 55 | 21 |
| 10 | 28 | 56 | 22 |
| 11 | 20 | 57 | 28 |
| 12 | 16 | 58 | 16 |
| 13 | 21 | 59 | 18 |
| 14 | 24 | 6 | 27 |
| 15 | 16 | 60 | 19 |
| 16 | 19 | 61 | 21 |
| 17 | 19 | 62 | 26 |
| 18 | 15 | 63 | 28 |
| 19 | 22 | 64 | 16 |
| 2 | 23 | 65 | 17 |
| 20 | 19 | 66 | 29 |
| 21 | 19 | 67 | 20 |
| 22 | 25 | 68 | 22 |
| 23 | 16 | 69 | 26 |
| 24 | 22 | 7 | 24 |
| 25 | 27 | 70 | 25 |
| 26 | 24 | 71 | 17 |
| 27 | 18 | 72 | 17 |
| 28 | 18 | 73 | 24 |
| 29 | 16 | 74 | 19 |
| 3 | 27 | 75 | 16 |
| 30 | 24 | 76 | 21 |
| 31 | 25 | 77 | 25 |
| 32 | 23 | 78 | 27 |
| 33 | 25 | 79 | 22 |
| 34 | 20 | 8 | 23 |
| 35 | 20 | 80 | 26 |
| 36 | 27 | 81 | 20 |
| 37 | 19 | 82 | 25 |
| 38 | 17 | 83 | 22 |
| 39 | 25 | 84 | 21 |
| 4 | 17 | 85 | 28 |
| 40 | 27 | 86 | 17 |
| 41 | 23 | 87 | 15 |
| 42 | 26 | 88 | 25 |
| 43 | 23 | 89 | 18 |
| 44 | 22 | 9 | 25 |
| 45 | 29 | 90 | 28 |
| 46 | 15 | 91 | 19 |
| 47 | 20 | 92 | 26 |
| 48 | 15 | 93 | 20 |
| 49 | 26 | 94 | 29 |
| 5 | 18 | 95 | 16 |
| 50 | 15 | 96 | 21 |
| 51 | 18 | 97 | 17 |
| 52 | 21 | 98 | 17 |
| 53 | 24 | 99 | 15 |

（5）各玩家平均评分

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 英雄数量 | 编号 | 英雄数量 |
| 0 | 50.2 | 54 | 52 |
| 1 | 50.85 | 55 | 50.37 |
| 10 | 51.96 | 56 | 51.92 |
| 11 | 52.03 | 57 | 50.48 |
| 12 | 51.32 | 58 | 51.11 |
| 13 | 53.71 | 59 | 49.23 |
| 14 | 52.53 | 6 | 52.89 |
| 15 | 53.75 | 60 | 50.02 |
| 16 | 49.19 | 61 | 52.73 |
| 17 | 49.96 | 62 | 46.94 |
| 18 | 50.33 | 63 | 46.39 |
| 19 | 51.39 | 64 | 52.05 |
| 2 | 51.03 | 65 | 48.35 |
| 20 | 49.67 | 66 | 52.1 |
| 21 | 44.99 | 67 | 50.39 |
| 22 | 49.77 | 68 | 54.93 |
| 23 | 49.26 | 69 | 51.79 |
| 24 | 49.99 | 7 | 48.73 |
| 25 | 51.71 | 70 | 47.5 |
| 26 | 47.36 | 71 | 53.03 |
| 27 | 49.24 | 72 | 43.92 |
| 28 | 49.39 | 73 | 50.37 |
| 29 | 47.06 | 74 | 46.72 |
| 3 | 54.02 | 75 | 51.15 |
| 30 | 52.52 | 76 | 53.27 |
| 31 | 50.7 | 77 | 51.1 |
| 32 | 53.82 | 78 | 49.92 |
| 33 | 49.09 | 79 | 48.52 |
| 34 | 51.28 | 8 | 48.26 |
| 35 | 50.86 | 80 | 51.61 |
| 36 | 47.15 | 81 | 55.43 |
| 37 | 46.62 | 82 | 48.39 |
| 38 | 54.33 | 83 | 51.83 |
| 39 | 52.39 | 84 | 49.67 |
| 4 | 50.91 | 85 | 53.8 |
| 40 | 55.63 | 86 | 48.67 |
| 41 | 50.87 | 87 | 52.09 |
| 42 | 47.83 | 88 | 47.68 |
| 43 | 49.17 | 89 | 52.21 |
| 44 | 48.54 | 9 | 52.35 |
| 45 | 51.76 | 90 | 51.84 |
| 46 | 51.32 | 91 | 51.23 |
| 47 | 49.49 | 92 | 47.99 |
| 48 | 51.28 | 93 | 50.32 |
| 49 | 48.99 | 94 | 51.5 |
| 5 | 47.51 | 95 | 54.09 |
| 50 | 54.41 | 96 | 50.41 |
| 51 | 48.95 | 97 | 53.02 |
| 52 | 51.17 | 98 | 47.18 |
| 53 | 51.57 | 99 | 51.52 |

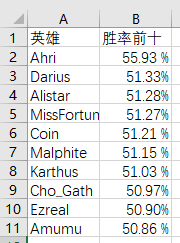
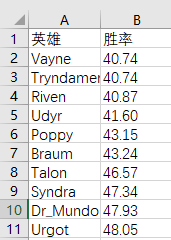
（6）各玩家kd比

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 英雄数量 | 编号 | 英雄数量 |
| 0 | 1.95 | 54 | 2.12 |
| 1 | 2.31 | 55 | 2.23 |
| 10 | 2.4 | 56 | 2.14 |
| 11 | 2.16 | 57 | 2.25 |
| 12 | 2.06 | 58 | 2.47 |
| 13 | 2.98 | 59 | 1.92 |
| 14 | 2.43 | 6 | 2.21 |
| 15 | 2.84 | 60 | 2.12 |
| 16 | 2.3 | 61 | 2.78 |
| 17 | 2.02 | 62 | 1.85 |
| 18 | 2.11 | 63 | 1.68 |
| 19 | 2.5 | 64 | 2.08 |
| 2 | 2.12 | 65 | 2.12 |
| 20 | 1.84 | 66 | 2.11 |
| 21 | 1.32 | 67 | 2 |
| 22 | 2.09 | 68 | 3.09 |
| 23 | 1.78 | 69 | 2.21 |
| 24 | 1.89 | 7 | 2.05 |
| 25 | 2.01 | 70 | 1.68 |
| 26 | 1.93 | 71 | 2.75 |
| 27 | 2.12 | 72 | 1.27 |
| 28 | 2.19 | 73 | 2.06 |
| 29 | 1.87 | 74 | 1.67 |
| 3 | 2.91 | 75 | 2.2 |
| 30 | 2.27 | 76 | 2.33 |
| 31 | 2.63 | 77 | 1.86 |
| 32 | 2.31 | 78 | 2.05 |
| 33 | 2.01 | 79 | 2.12 |
| 34 | 2.54 | 8 | 2.17 |
| 35 | 2.45 | 80 | 1.94 |
| 36 | 1.94 | 81 | 3.22 |
| 37 | 1.61 | 82 | 1.86 |
| 38 | 2.95 | 83 | 2.44 |
| 39 | 2.51 | 84 | 2.08 |
| 4 | 2.1 | 85 | 2.38 |
| 40 | 3 | 86 | 2.27 |
| 41 | 2.5 | 87 | 2.73 |
| 42 | 1.76 | 88 | 2.2 |
| 43 | 2.18 | 89 | 2.09 |
| 44 | 1.94 | 9 | 2.57 |
| 45 | 2.44 | 90 | 2.26 |
| 46 | 2.48 | 91 | 2.72 |
| 47 | 1.93 | 92 | 1.98 |
| 48 | 2.42 | 93 | 2.03 |
| 49 | 2.16 | 94 | 2.4 |
| 5 | 2.05 | 95 | 3.08 |
| 50 | 2.57 | 96 | 2.15 |
| 51 | 2.28 | 97 | 2.59 |
| 52 | 1.68 | 98 | 1.64 |
| 53 | 1.87 | 99 | 2.11 |

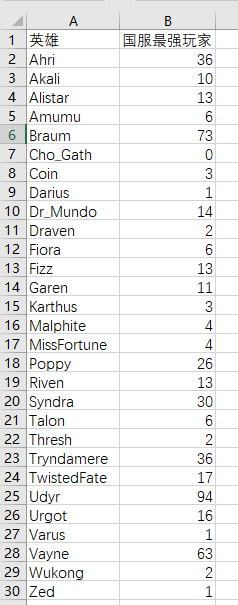
（7）按照id分区

此部分在模块4中展示

（8）英雄胜率排行（前十名与后十名）

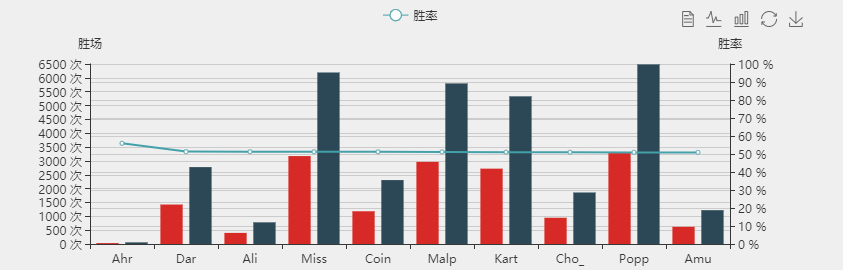
 

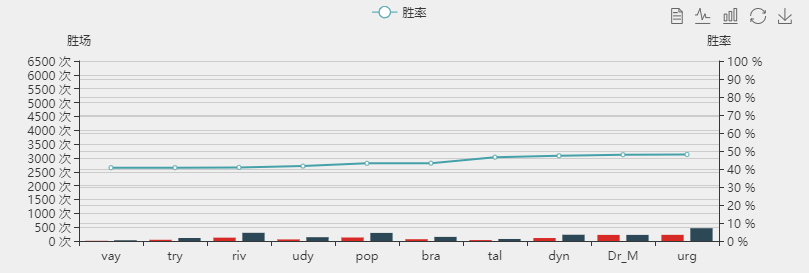
1. 最强玩家 大神榜



1. 数据分析

（1）由胜率前十的英雄/后十英雄制成图表对照





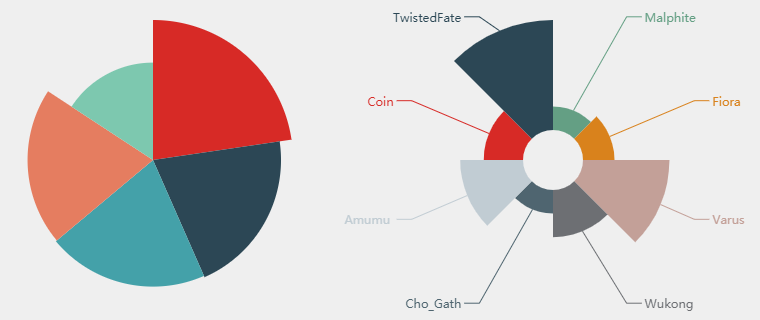
其中，蓝色柱体表示使用次数，红色柱体表示胜场，蓝色折线表示胜率。

结论1：胜率高的英雄不一定使用次数多，不一定热门

结论2：胜率低的英雄一定使用次数较少

结论3：版本强势英雄确实存在，胜率高低与英雄类型由相关性，这可能与属性、装备因素有关。

（2）调查对象：5号玩家，该玩家KD为2.05，属于中游水平





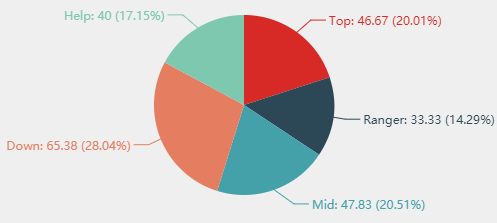
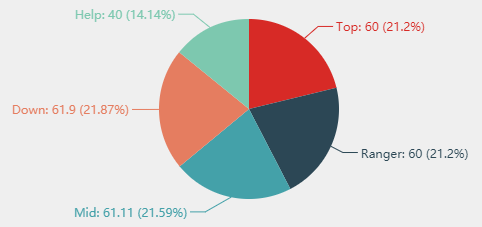
左图为该玩家各个位置的胜率，右图为该玩家胜率最高的英雄

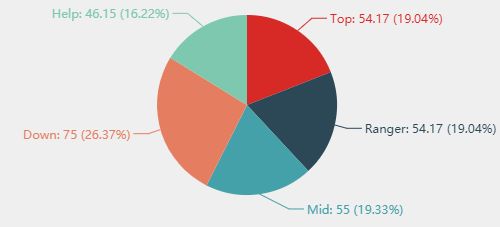
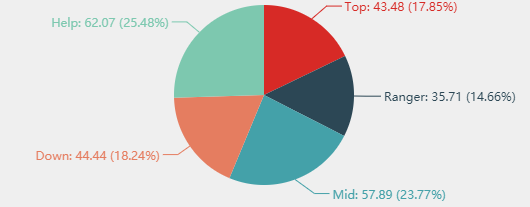
结论4：各个位置的胜率与胜率较高英雄职业有绝对的相关性

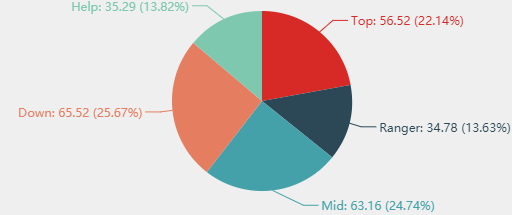
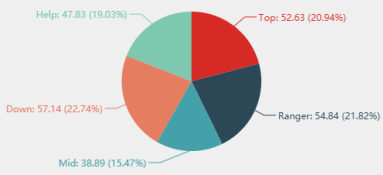
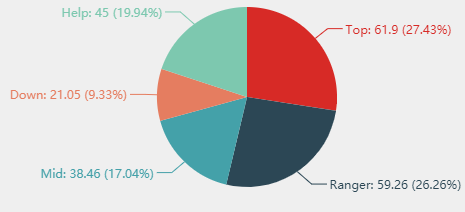
结论5：英雄职业与分路存在较大的相关性

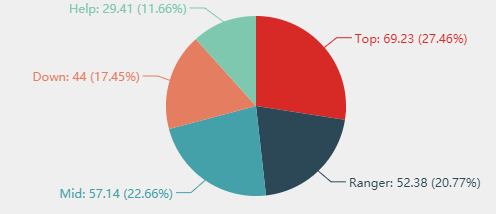
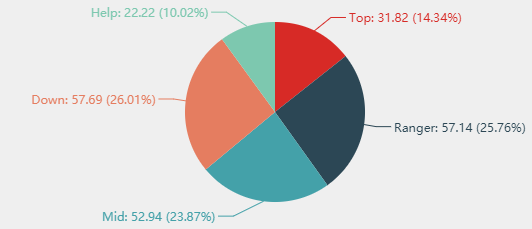
（3）











这是10位玩家的KD & 选择分路情况统计，其中，前五图对应KD前五名，后五图为KD后五名，各种颜色表示在第一行有注明。

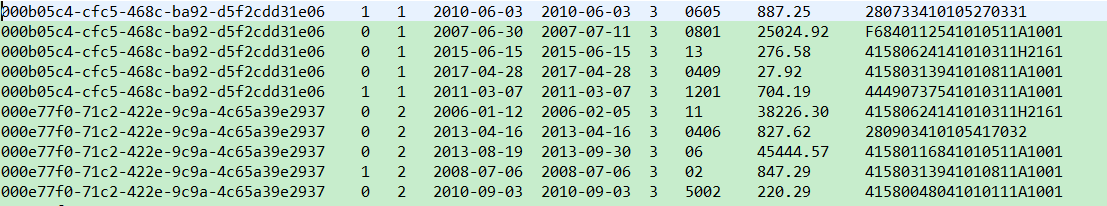
结论6：KD与分路并没有直接的线性关联

结论7：由结论5、6，英雄职业与KD没有直接的线性关联

结论8：各个位置胜率与KD没有直接关联。此结论可有结论4、6推出，也可以由图得出，护卫佐证。

四、基于Hadoop、Hive的电商交易日志分析

**1、项目基本介绍**

原始数据SHOP\_LOG

0—用户名 1—付款状态（1代表下单，0代表取消订单） 2—性别

Hive建表代码：create table shop\_log(line string);

load data inpath '/shop.log'

overwrite into table shop\_log;

create table shop\_test as

select order\_id ,user\_id,province,city\_id,way\_number,channel,login,logout,money,status

from (

select split(line,'\\s+')[0] as order\_id,split(line,'\\s+')[1] as user\_id,split(line,'\\s+')[2] as province,split(line,'\\s+')[3] as city\_id,split(line,'\\s+')[4] as way\_number,split(line,'\\s+')[5] as channel,split(line,'\\s+')[6] as login,split(line,'\\s+')[7] as logout,split(line,'\\s+')[8] as money,split(line,'\\s+')[9] as status

from shop\_log

)a

;

**2、需求详解**

1）统计各用户累计有效消费金额（付款状态为1），以及累计购买次数，进行统计和排名（累计消费金额>>累计消费次数）。取前二十。

UserInfo.java UserRanking.java

累计用户：9667个，前二十如下：



代码思路：map端读入数据后，key为用户ID，value为后面的所有信息，经过reduce端对其聚合和统计输出后，在UserRankin中通过compare函数对其消费金额和累计消费次数进行排序输出前20。

Hive代码：

//统计

drop table if exists shop\_count;

create table if not exists shop\_count as

select user\_id as user\_id,SUM(cast(money as double)) as money\_count,count(1) as total

from shop\_test where status="1"

group by user\_id;

//降序取前20

select \* from shop\_count order by money\_count,total desc limit 20;

2) 按照下单的时间年份统计每年的销售量，销售额并对这15年每年的销售情况做排名（消费金额>>消费次数）

YearsSales.java YearsSalesRanking.java



代码思路：map端读入数据后，key为下单时间，value为后面的所有信息，经过reduce端对其聚合和统计输出后，在YearsSalesRankin中通过compare函数对其消费金额和累计消费次数进行排序输出。

Hive代码：

drop table if exists year\_count;

create table if not exists year\_count as

select substr(logout,0,4) as year,SUM(cast(money as double)) as money\_count,count(1) as total

from shop\_test where status="1"

group by substr(logout,0,4);

//降序取前20

select \* from year\_count order by money\_count,total desc;

3）统计各省份的销售量，销售额。对这10个省的销售情况做排名（消费金额>>消费次数）。

LocationSales.java LocationSalesRanking.java



代码思路：map端读入数据后，key为省份代号，value为后面的所有信息，经过reduce端对其聚合和统计输出后，在LocationSalesRankin中通过compare函数对其消费金额和累计消费次数进行排序输出。

Hive代码：

drop table if exists province\_count;

create table if not exists province\_count as

select substr(province,0,1) as province,SUM(cast(money as double)) as money\_count,count(1) as total

from shop\_test where status="1"

group by substr(province,0,1);

//降序取前20

select \* from province\_count order by money\_count,total desc limit 20;

4）统计各月份的销售量，销售额。对十二个月的销售情况做排名（消费金额>>消费次数）。

MonthsSales.java MonthsSalesRanking.java



代码思路：map端读入数据后，key为月份代号，value为后面的所有信息，经过reduce端对其聚合和统计输出后，在MonthsSalesRankin中通过compare函数对其消费金额和累计消费次数进行排序输出。

Hive代码:

drop table if exists month\_count;

create table if not exists month\_count as

select split(logout,'-')[1] as month,SUM(cast(money as double)) as money\_count,count(1) as total

from shop\_test where status="1"

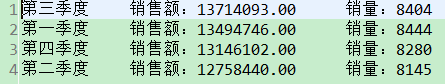
group by split(logout,'-')[1];

//降序取前20

select \* from month\_count order by money\_count,total desc;

5）统计各个季度(共四个季度)的销售量，销售额。对各个季度的销售情况做排名（消费金额>>消费次数）。

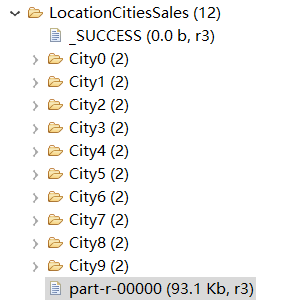
QuarterYearsSale.java QuarterYearsSalesRanking.java



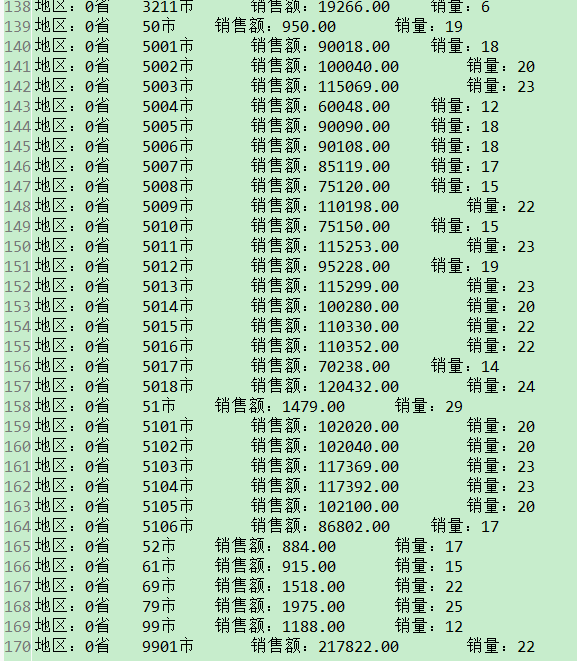
代码思路：用4中的结果文件，map端读入数据后，key为月份代号，value为后面的所有信息，经过reduce端对其聚合和统计输出后，在QuarterYearsSalesRanking中通过compare函数对其消费金额和累计消费次数进行排序输出。

6）统计各市区的销售量，销售额。

LocationCitiesSales.java



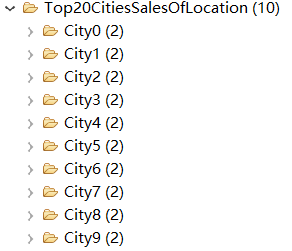
展示City0的市区如下：（节选）



代码思路： 先对省份分区，经过map端读入数据后，key为市区代号，value为后面的所有信息，经过reduce端对其聚合和统计后输出。

7）利用已经按照省份地区进行分区过的数据，通过排名得到每个城市中销售额最多的20个市区。

Top20CitiesSalesOfLocation.java



展示City0中销售额最多的20个市区如下：

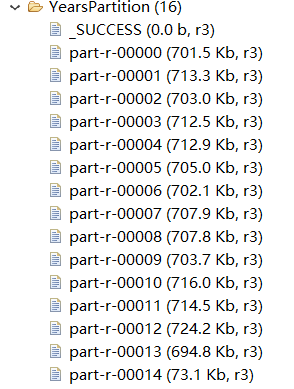


代码思路： 利用省份分区后的结果，经过map端读入数据后，key为市区代号，value为后面的所有信息，经过compare端对其排名后输出。

8）将数据按照购买年份进行分区, 分成15份。统计这15年每一年里各个省份的销售额，销售量。对每一年（共15年）各个省份的消费情况进行排名（消费金额>>消费次数）。

EachYearCitiesSalesPartition.java EachYearCitiesSales.java

EachYearCitiesSalesRanking.java

展示2004年的各省份消费排名如下：



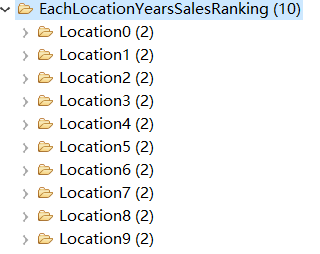
代码思路：将原文件进行年份的分区后，map端读入数据，key为每个省份，value为后面的信息，在reduce端对key聚合并进行统计和输出，然后在EachYearCitiesSalesRanking中利用compare函数进行排序和输出。

9）将数据按照省份地区进行分区。统计每个地区在每年的销售额，销售量。对每个地区（共10个）十五年来的消费情况进行排名（消费金额>>消费次数）。

EachLocationYearsSalesPartition.java EachLocationYearsSales.java

EachLocationYearsSalesRanking.java

代码思路：将数据按照省份进行分区，在分区后的每个文件里以年份作为key统计每年的销售额、销售量并进行排名。



展示9省的十五年来的消费情况进行排名如下：



10）统计男、女的销售额和销售量，并计算其比重。

MenAndWomenSales.java

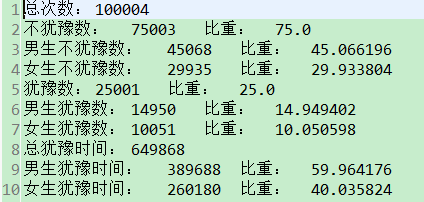
代码思路：在Map端将用户id作为key,其余部分作为value，在Reduce端确定性别，并累加计算男女的销售额和销售量。



11）统计多少人购买时无考虑时间(下单时间等于上线时间)，并计算男女在其中分别占有的比重。统计多少人购买时有考虑时间(下单时间等于上线时间)，并计算男女在其中分别占有的比重。统计男女考虑时间累计有多少，并求其分别所占的比重。

ThinkTime.java

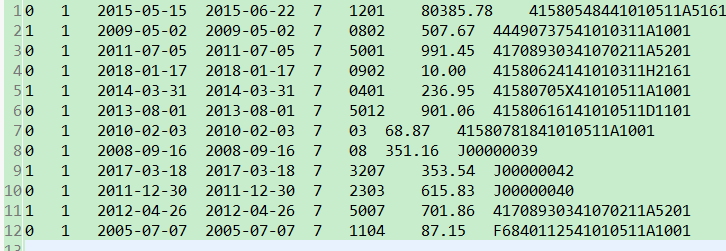
代码思路：通过计算下单时间和上线时间的差值，求出男女的考虑时间 。



12）传参数查询某一个ID的相关信息，比如0040ffc5-8954-4b28-96a1-21f5e8d274e8。通过查询到的某个ID分析他/她的每年的购买额和购买力。

SesrchIDInfo.java SearchIDSales.java

代码思路：在main函数里传参数，如果查询到这个ID，在REDUCE端进行购买额和购买力的计算。



同时，查询某个ID的情况，在HIVE里这样实现：

select \* from shop\_count where user\_id="41630758341020411A1001";

13）通过传参数展示每年、每月、每地区的销售量销售额，比如2016年，12月，5省。

YearMonthLocationSales.java

代码思路：在REDUCE端进行销售量销售额的累加，在main函数里传参数。



14）将所有订单按照金额进行分区，并分区统计下单与取消订单的数量。

MoneyRangePartition.java

代码思路：选取原始数据的金额部分，按照金额的大小进行分区，具体的界限如下。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **金额** | **<5000** | **5000-10000** | **10000-20000** | **20000-30000** | **30000-40000** | **40000-50000** | **>50000** |
| **总订单数** | 76208 | 1304 | 2437 | 2522 | 2446 | 2502 | 12585 |
| **下单** | 25338 | 451 | 820 | 848 | 783 | 843 | 4190 |
| **取消订单** | 50870 | 853 | 1617 | 1674 | 1663 | 1659 | 8395 |

15)将数据按照付款状态进行分区：

StatuPartition.java

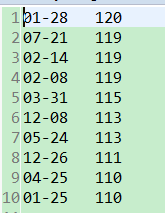
代码思路：PartitionPart里先将原始数据一行进行分割，付款状态为0的和1的进行分区。

|  |  |  |
| --- | --- | --- |
| **付款状态** | **下单** | **取消订单** |
| **数量** | 33273 | 66731 |

16）-1统计所有订单中一年中每天下单的数量，并得到下单数最多的十天。

DaySalesNum.java DaySalesNumTOP10.java

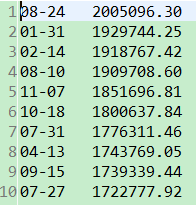
代码思路：MAP输出端将日期如2006-07-28用substring取07-28作为key,将one设置为value，在REDUCE端计算数量。



16)-2统计所有订单中一年中每天下单的金额，并得到销售额最多的十天。

DaySalesMoney.java DaySalesMoneyTOP10.java

代码思路：在MAP端将日期如2006-07-28用substring取07-28作为key，整行作为value,在REDUCE端计算金额。



17）在HIVE端统计平台的情况

drop table if exists channel\_count;

create table if not exists channel\_count as

select channel as channel,SUM(cast(money as double)) as money\_count,count(1) as total

from shop\_test where status="1"

group by channel;

select \* from channel\_count;

代码思路：在HIVE实现的时候，采用了原始的数据格式，

0-订单编号：order\_id

1-用户：id user\_id

2-省份：province

3-市区：city\_id

4-运单编号：way\_number

5-购买渠道：channel（1：pc，2：移动）

6-上线时间：login

7-下线时间：logout

8-交易金额：money

9-付款状态：status（0：未付款）

故针对购买的平台渠道进行统计。

18）在HIVE端统计无考虑时间的人数且为移动端购买。

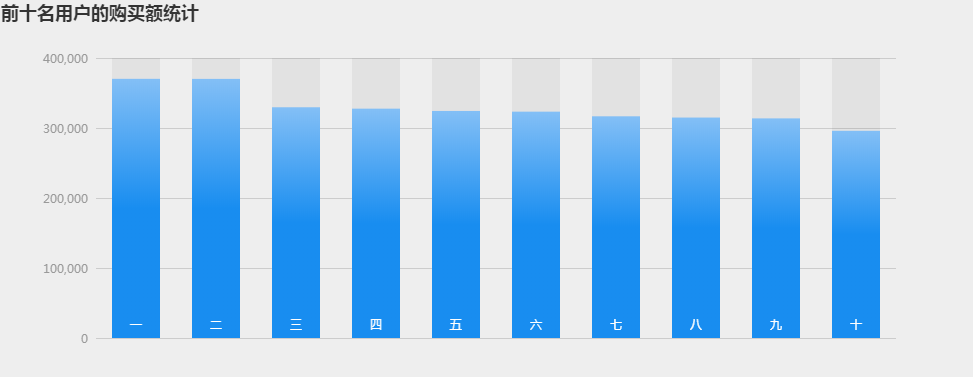
select count(1) from shop\_test where login=logout and channel="2";

result: 75003

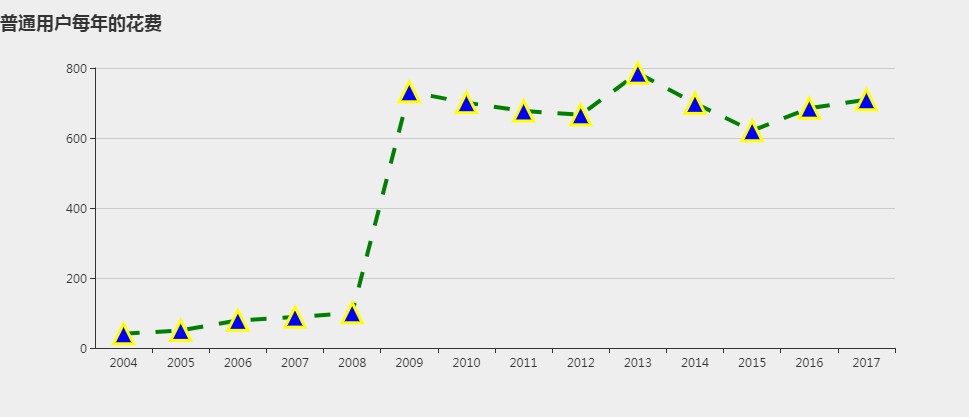
代码思路：无考虑时间：login=logout，移动端channel="2"

**3、结果分析**

1. **针对用户的购买情况分析**

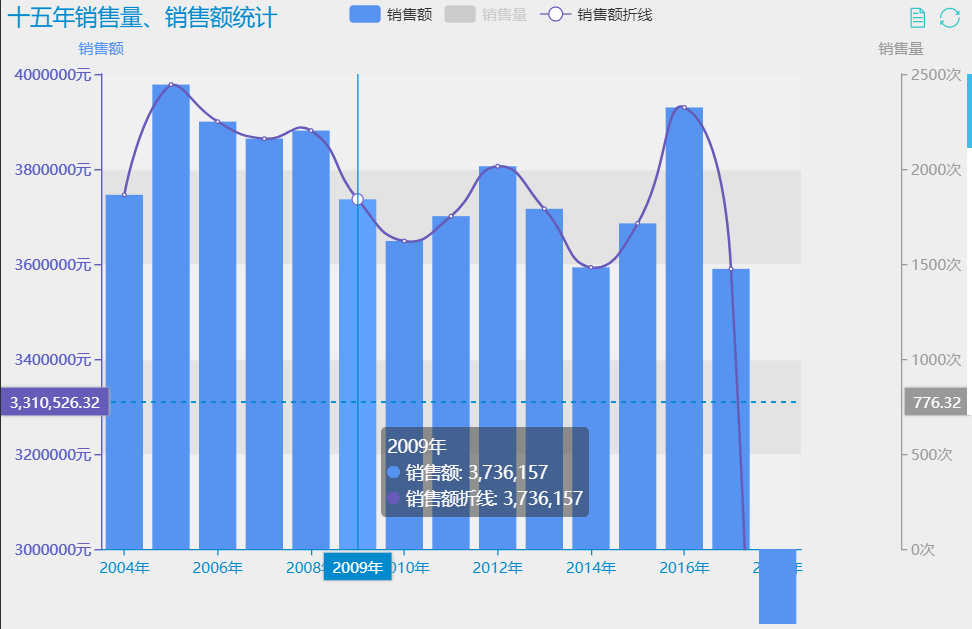


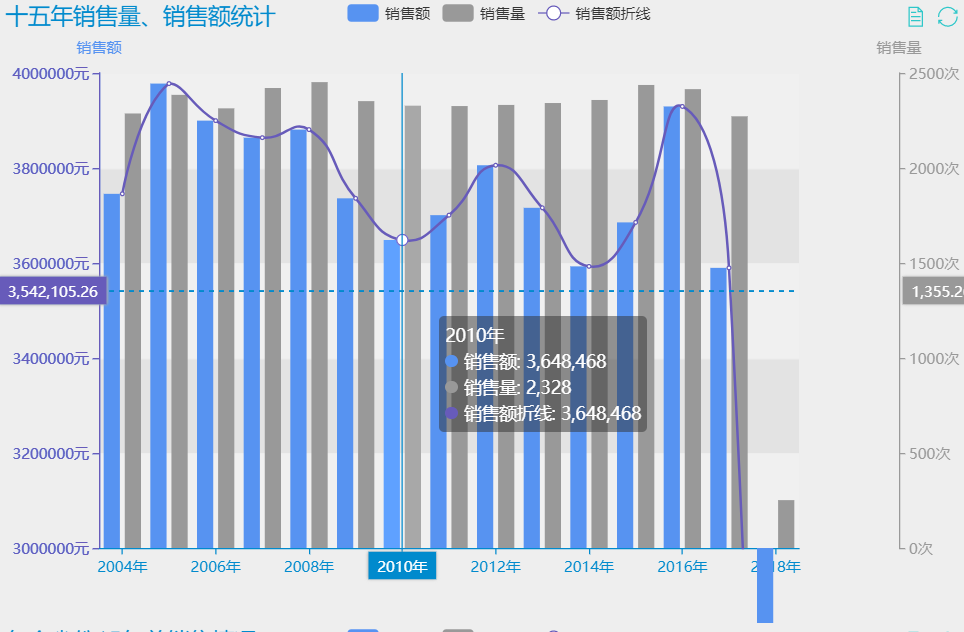
从数据中可以看出前十名用户的购买额基本都在三十万元以上，下一阶段我们要做的就是通过一些促销、优惠活动鼓励前十名继续消费，稳定消费额较高的用户的市场，从而带动整体的销售额。



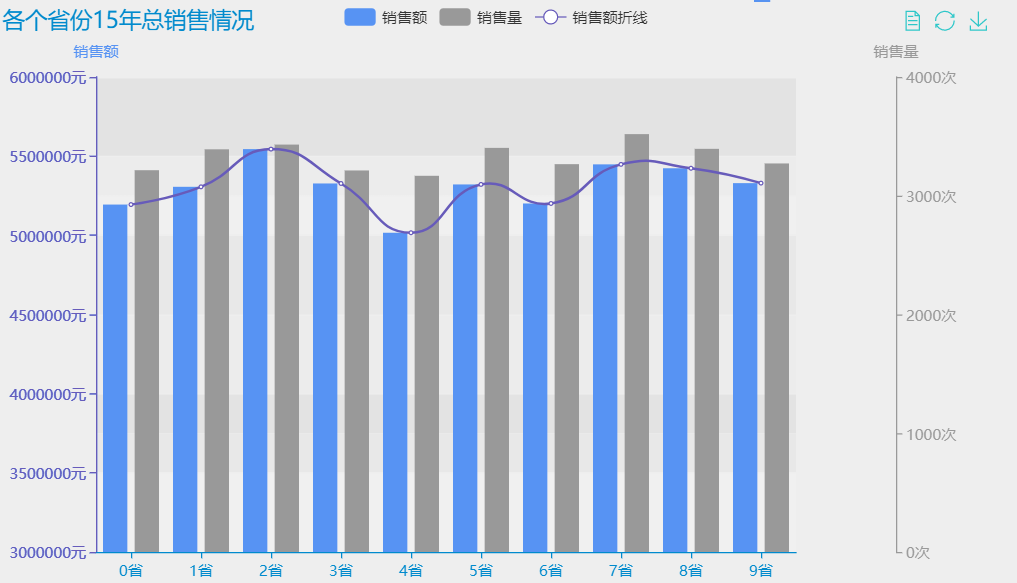
从第二个表格我们可以看出，从2009年以后普通用户的消费水平骤增，而且在近几年不断波动， 在2013年创历史新高，在2017年又呈上升趋势，说明用户对网上购物已经呈现疲态，我们要抓住此次时机用促销、 优惠或更新奇的方式吸引用户，以提高平均购买水平。

**2）年销售情况**

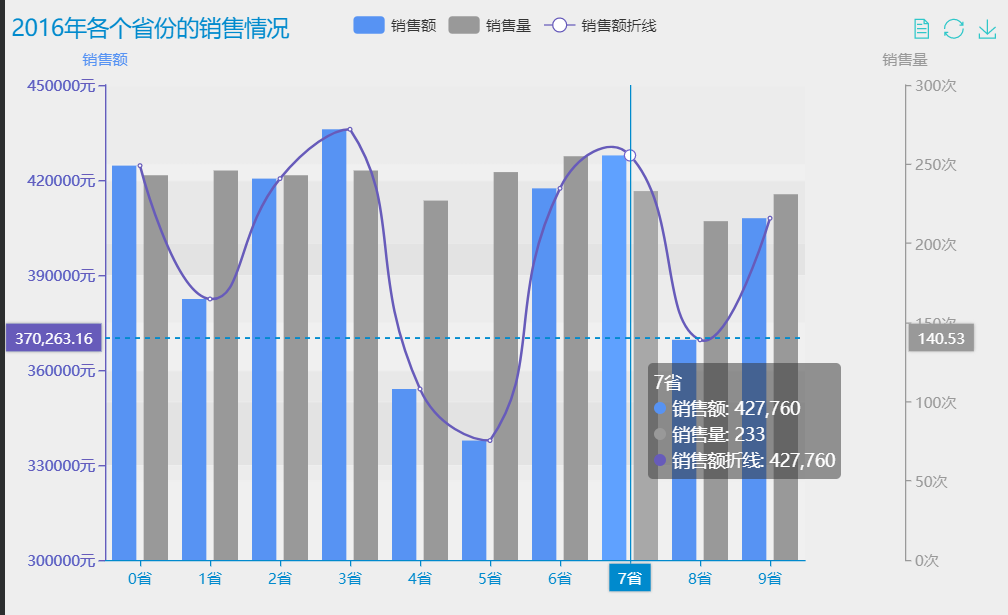




从表上数据可以看出，十五年来销售额起伏不定，但相对稳定在380万元。

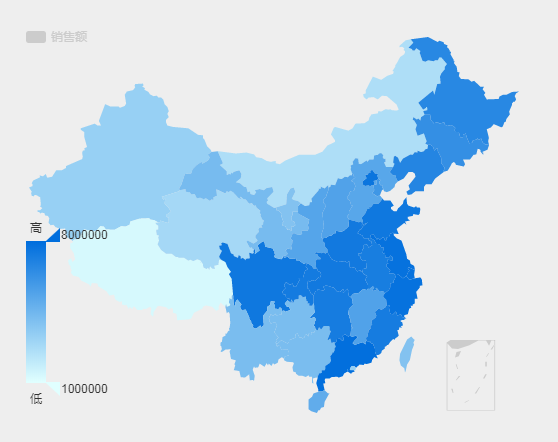


从这个表可以看出，十五年来十个省份销售均较良好，其中2省销售额最为突出。



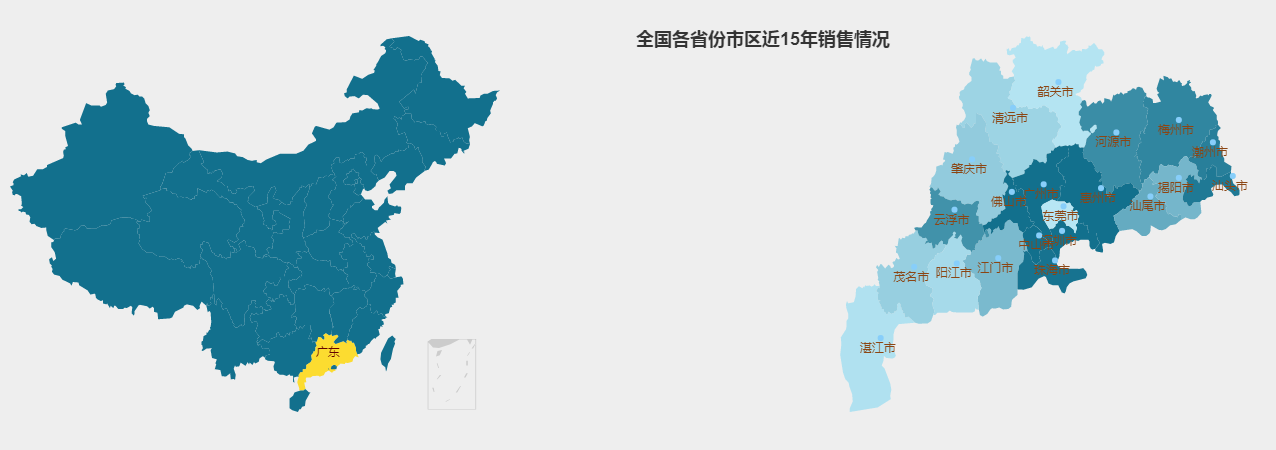
从这个表可以看出，2016年各个省份的销售参差不齐，其中3省与7省销售最好，1省、5省与8省销售最低。

**3）针对全国各省总销售情况分析**



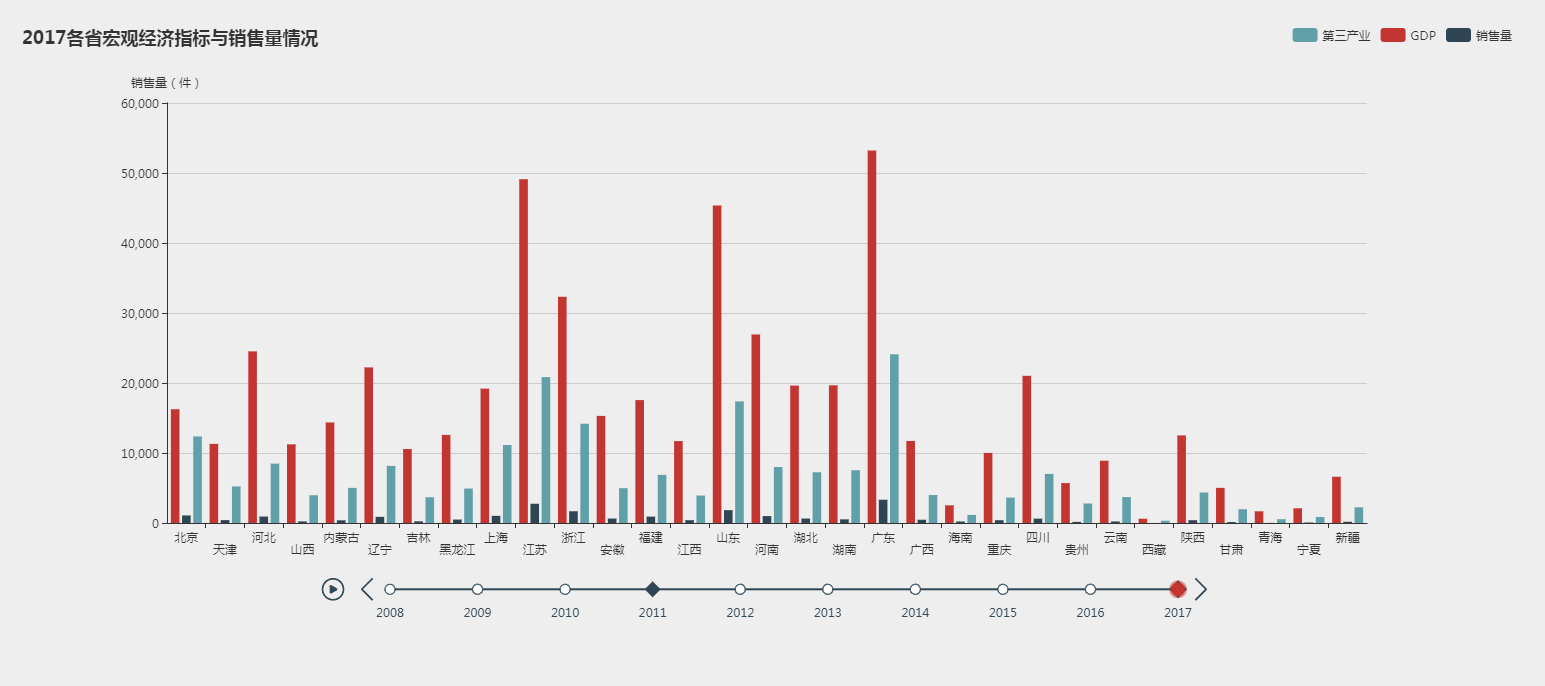
图中区域颜色的深浅分别代表销售额的多少，从图中我们可以看出沿海经济发达区和人口大省的销售额都较高，在以后的发展过程中，我们可以重点关注这几个省，利用其优势促进电子商务的发展。

而对排名较为靠后的省份，从地图上我们可以发现，基本分布在内陆交通不发达地区以及经济较为落后地区， 在以后的发展中，可以通过减免邮费，或加大促销力度来刺激其电子商务消费的发展。



对排名第一的广东省进行分析，我们可以看出消费能力和各市区的GDP有很大关系，以后可以通过对GDP不发达市区进行大力度的优惠活动从而刺激其消费力度的发展 进一步提升其省份的消费能力。

**4）针对全国各省份15年销售情况分析**

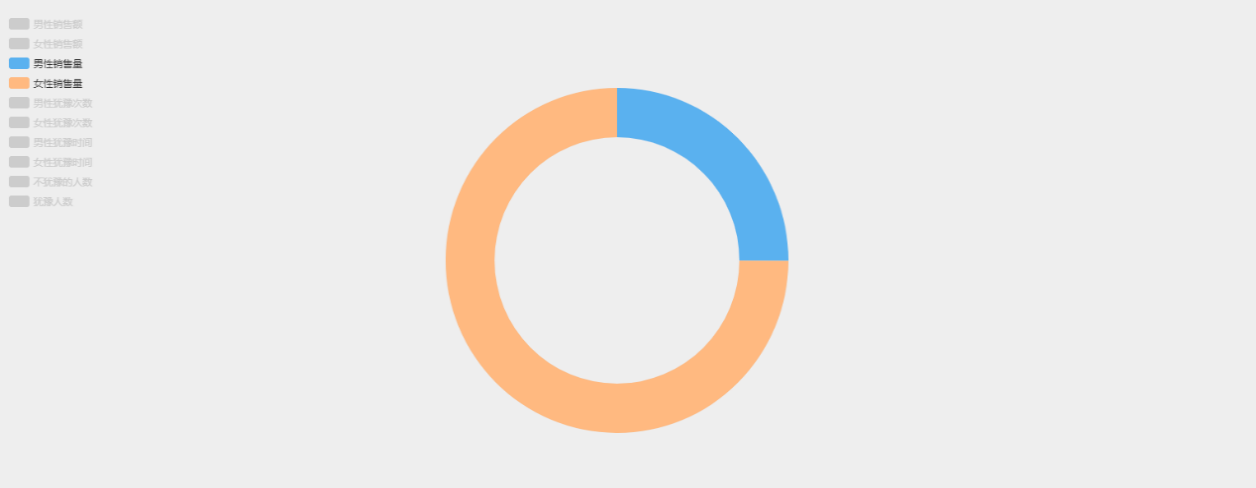


从上面的动态表，我们可以看出销售量和省份的GDP以及第三产业（这里指交通运输业）的发展有很大关系，但江苏省虽然GDP居于全国第二，但其销售量却并没有这么好的成绩，说明在以后的发展中我们应该加大对江苏省的宣传力度和促销力度， 从而刺激江苏地区消费者的购买热情。

**5）针对男性、女性的购买情况分析**

销售额



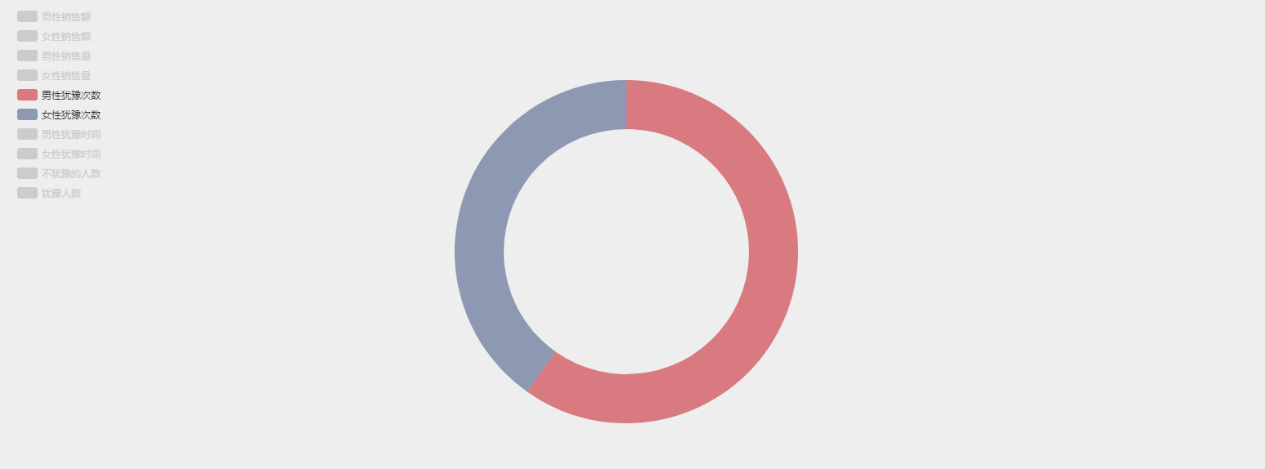
销售量  


对男、女销售额、销售量进行分析我们可以看出女性销售额和销售量都是男性的三倍，所以在以后的发展中，我们要通过加大宣传力度和促销力度保证女性消费市场的稳定发展同时要在男性用品方面加大宣传和推广，刺激男性的购买欲，从而扩大男性消费者的市场。

考虑时间

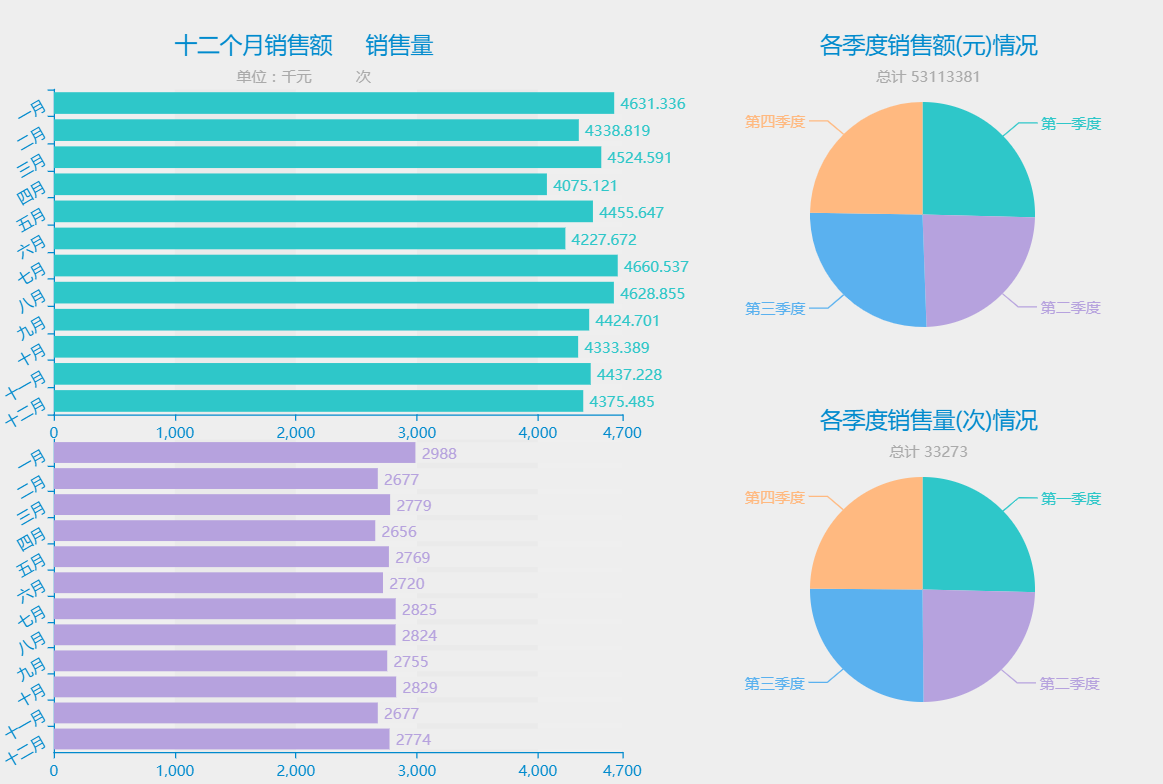


考虑次数



从考虑情况来看，男性的犹豫次数和犹豫时间都比女性多，所以我们应该加大对男性的促销力度，减少其犹豫时间和犹豫次数，提高其有效的购买力。

**6）月季度销售情况**

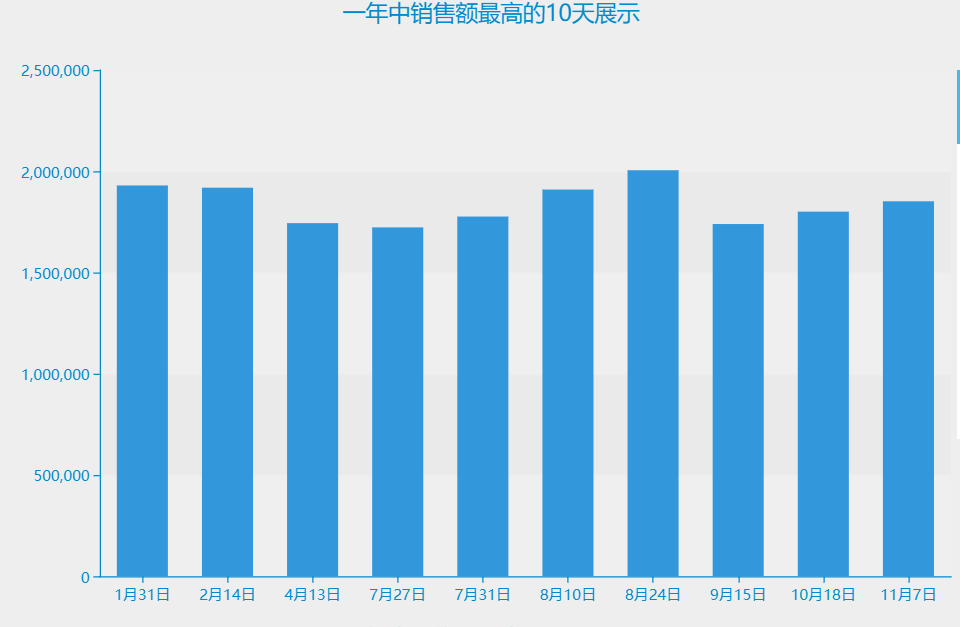


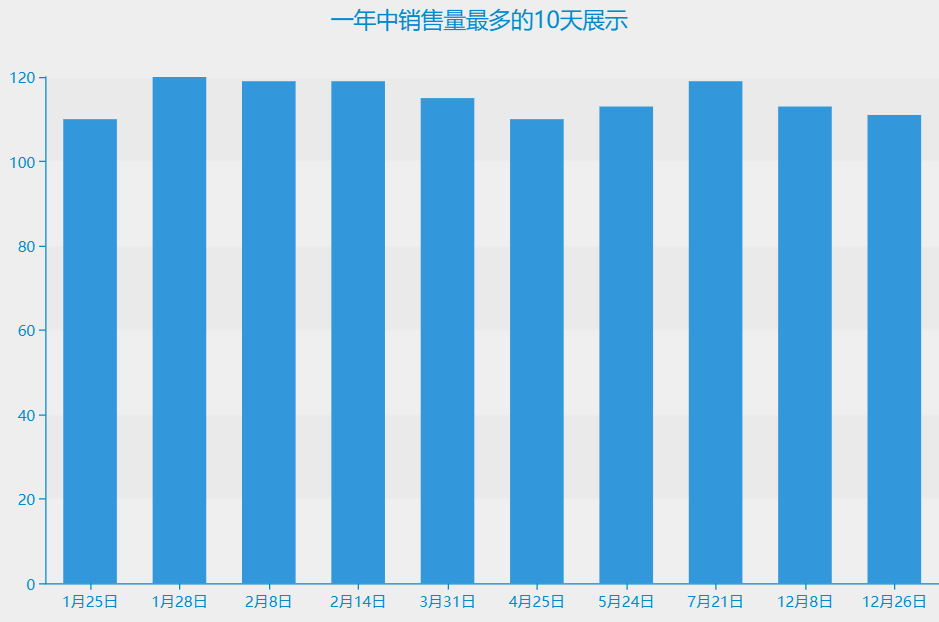
从图表分析可得：

1、从销售额看，十二个月中七月销售最好，四月销售较差，四个季度销售额相当。

2、从销售量看，十二个月中一月销售最好，四个季度销售相当。

**7）十佳销售日**





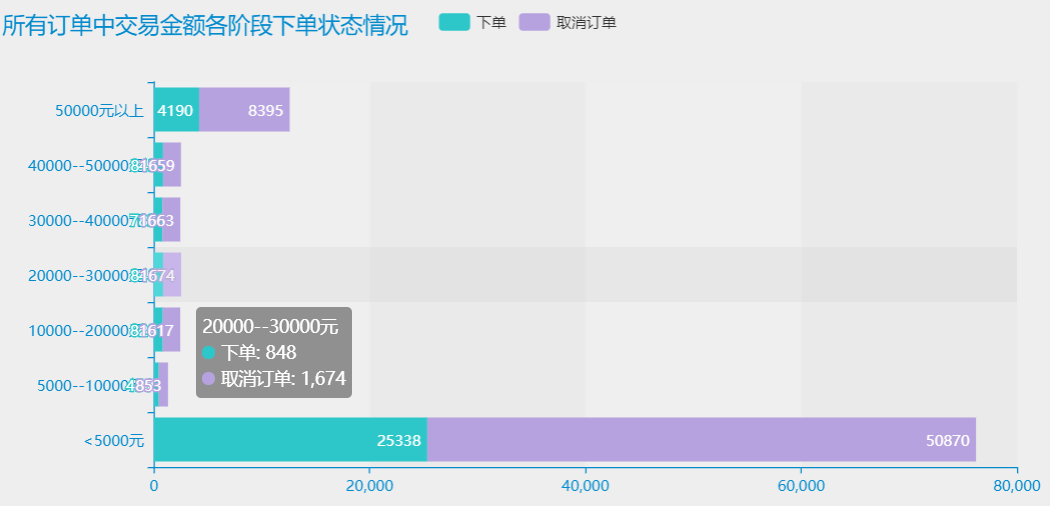
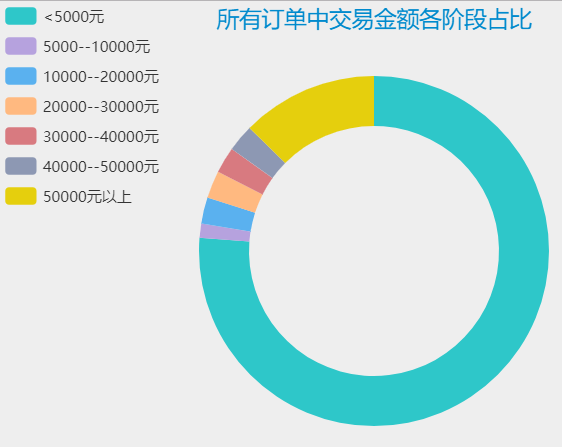
从这两个图表分析可得：

1、从表1来看，一年中销售额最高的十天有：1月31日，2月14日，4月13日，7月27日，7月31日，8月10日，8月24日，9月15日，10月18日，11月7日。

2、从表2来看，一年中销售量最好的十天有：1月25日，1月28日，2月8日，2月14日，3月31日，4月25日，5月24日，7月21日，12月8日，12月26日。

3、综合来看，一年中就会锁定销售额、销售量较好的十佳销售日范围。比如2月14日，可能是情人节，销售额与销售量都比较好。

**8）订单金额与状态**



从图表分析可得：

1、从表1来看，所有订单中交易金额小于5000元的订单占比最高，达76.21%，50000元以上的订单次之。

1、从表2来看，所有订单中取消订单的占比较高，达66.73%，下单的占比为33.27%。

3、从表3来看，所有订单中交易金额各阶段，同样是取消订单的占比超过50%。由此可见，取消订单与交易金额的大小并无直接关系。