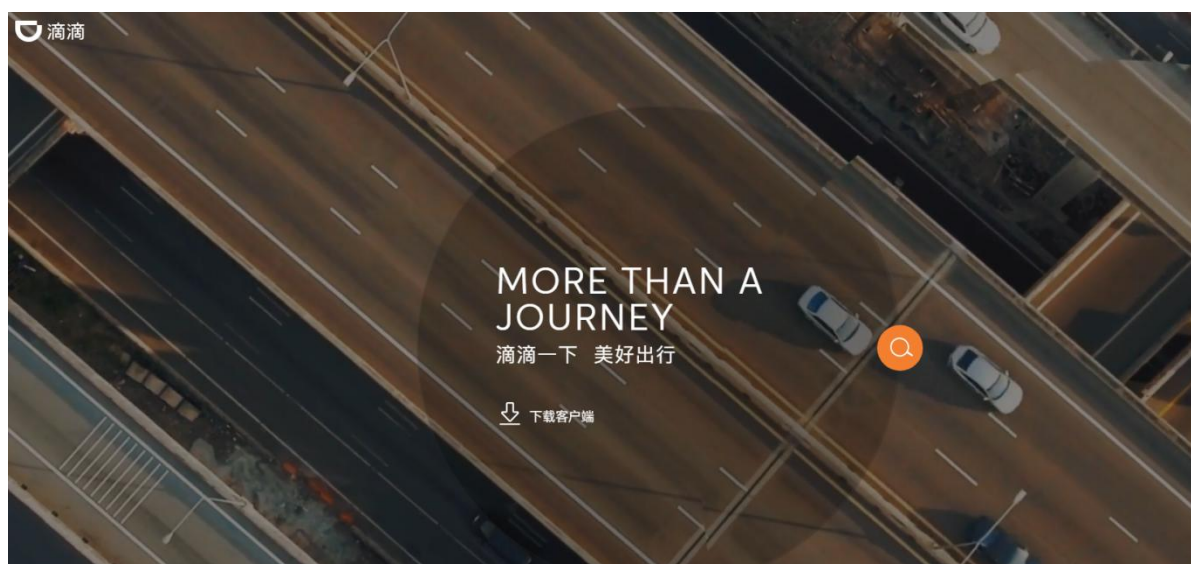


第三章 滴滴出行数据分析

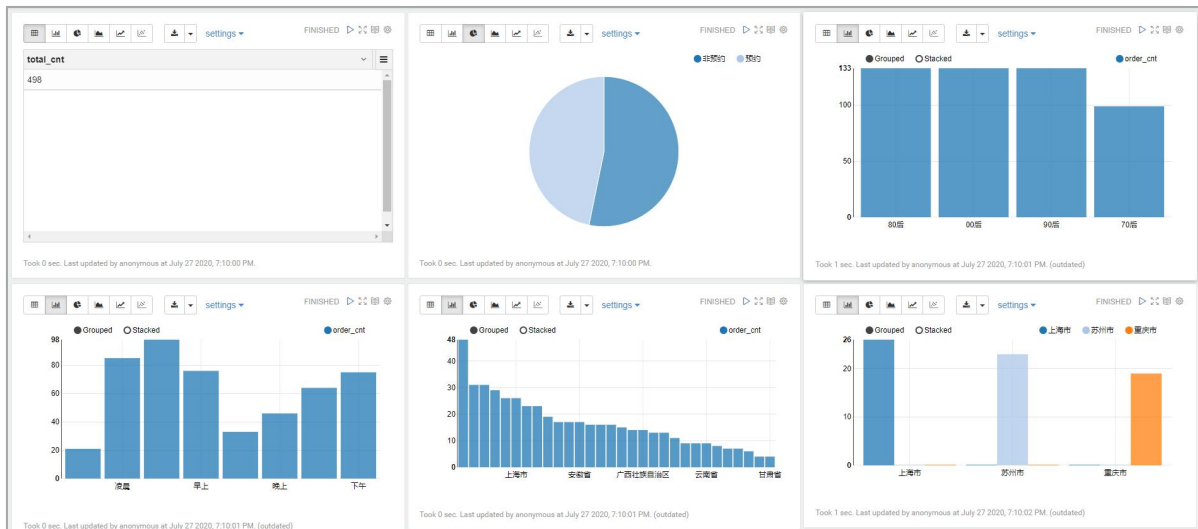
1. 业务背景

1.1 业务介绍



滴滴拥有超过4.5亿用户，在中国400多个城市开展服务，每天的订单量高达2500W，每天要处理的数据量4500TB。仅仅在北京，工作日的早高峰一分钟内就会有超过1600人在使用滴滴打车。通过对这些数据进行分析，了解到不同区域、不同时段运营情况。通过这些出行大数据，还可以看到不同城市的教育、医疗资源的分布，长期观察对城市经济、社会资源的发展、变迁情况，有非常有研究价值。

本次的案例将某出行打车的日志数据来进行数据分析，例如：我们需要统计某一天订单量是多少、预约订单与非预约订单的占比是多少、不同时段订单占比等。最终效果如下：



通过本案例，我们将使用以下技术来进行数据分析：

1. HDFS
2. Hive
3. Spark SQL
4. Zeppelin

1.2 架构图

要进行大规模数据分析，我们要考虑几个问题：

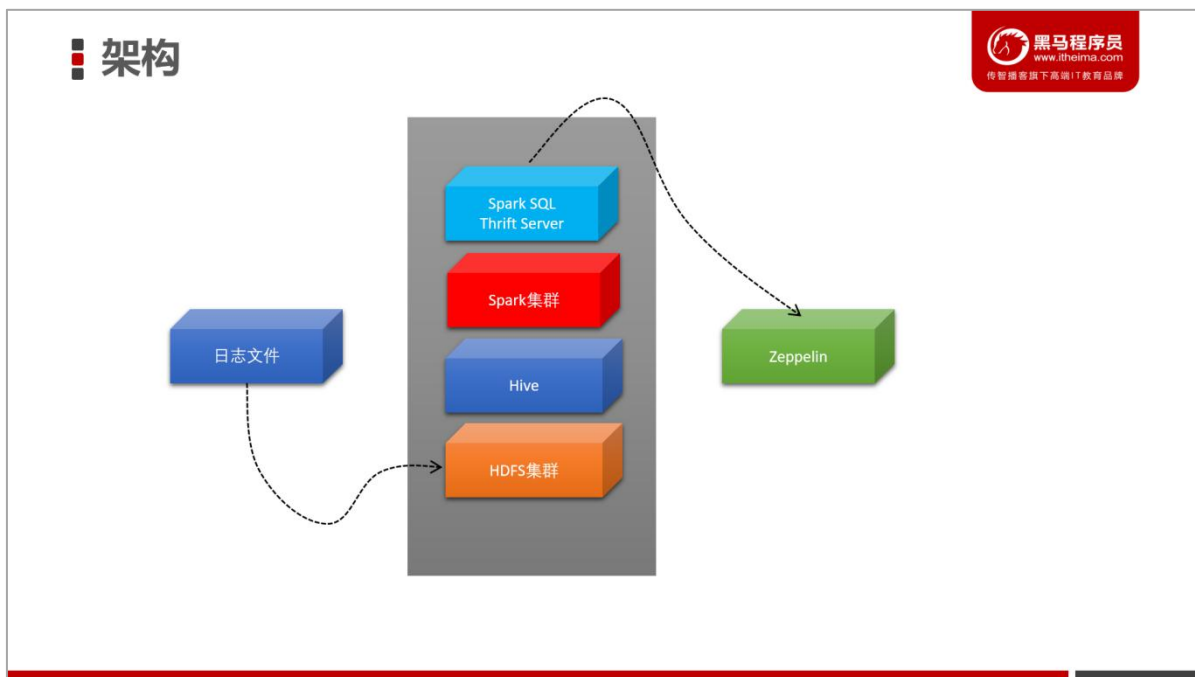
1. 打车的用户量非常庞大，数以亿记的用户将会有海量的数据需要存储。如何保存这些数据呢？
2. 为了方便对这些大规模数据进行处理、分析，我们如何建立数据模型，方便进行业务分析呢？
3. 亿级的数据如何保证效率，效率分析？
4. 数据分析的结果，应该以更易懂的方式呈现出现，如何展示这些数据？

要解决这些问题，我们需要设计一套大数据架构来解决上述问题。

解决方案：

1. 用户打车的订单数据非常庞大。所以我们需要选择一个大规模数据的分布式文件系统来存储这些日志文件，此处，我们基于Hadoop的HDFS文件系统来存储数据。
2. 为了方便进行数据分析，我们要将这些日志文件的数据映射为一张一张的表，所以，我们基于Hive来构建数据仓库。所有的数据，都会在Hive下来几种进行管理。为了提高数据处理的性能。
3. 我们将基于Spark引擎来进行数据开发，所有的应用程序都将运行在Spark集群上，这样可以保证数据被高性能地处理。

4. 我们将使用Zeppelin来快速将数据进行可视化展示。



2. 日志数据集介绍

2.1 日志数据文件

我们要处理的数据都是一些文本日志，例如：以下就是一部门用户打车的日志文件。

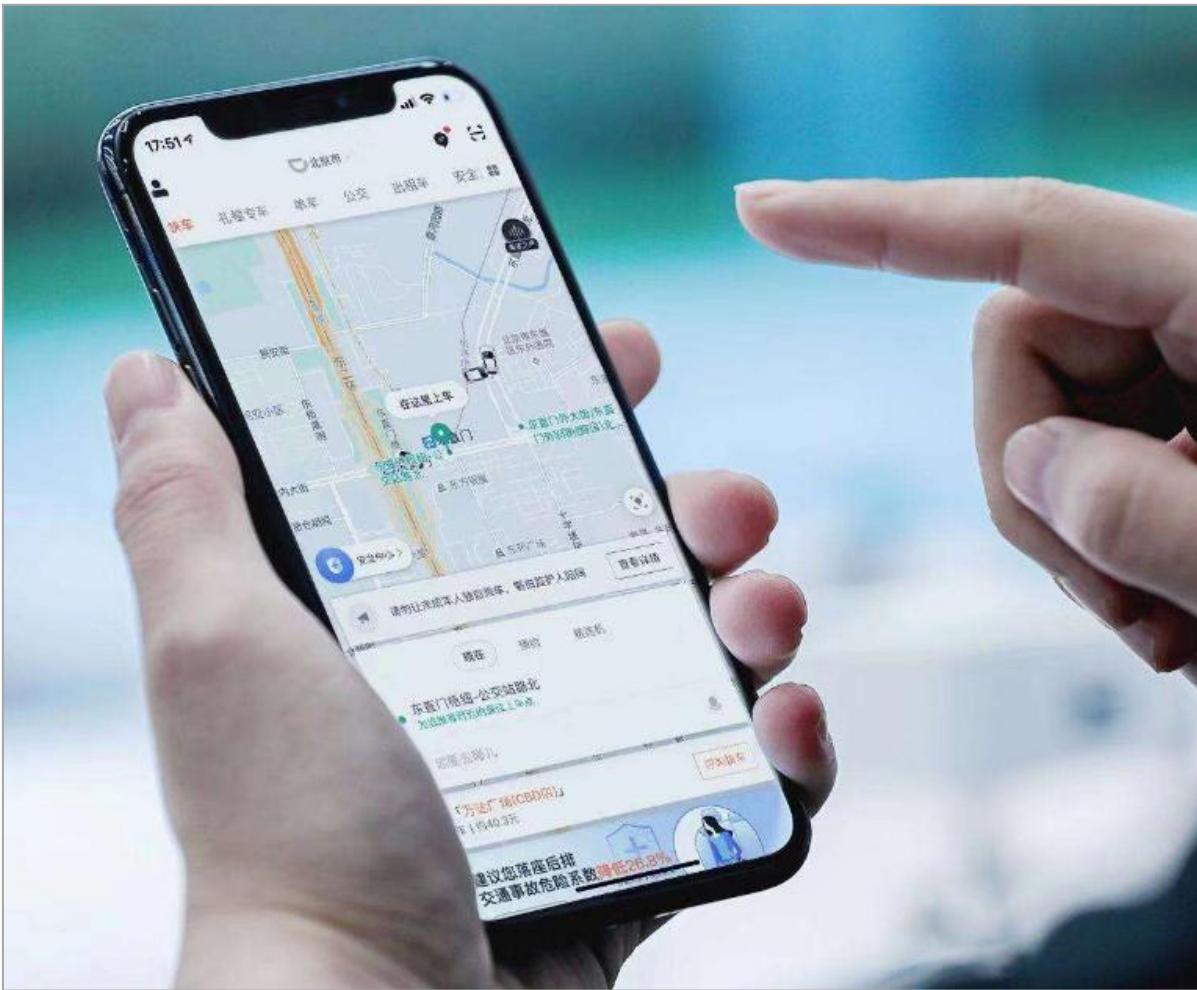
```
b05b0034cba34ad4a707b4e67f681c71,15152042581,109.348825,36.068516, 陕西省 , 延安市,78.2,男,软件工程,70后,4,1,2020-4-12 20:54,0,,2020-4-12 20:06
23b60a8ff11342fcadab3a397356ba33,15152049352,110.231895,36.426178, 陕西省 , 延安市,19.5,女,金融,80后,3,0,,0,,2020-4-12 4:04
1db33366c0e84f248ade1efba0bb9227,13905224124,115.23596,38.652724, 河北省 , 保定市,13.7,男,金融,90后,7,1,2020-4-12 10:10,0,,2020-4-12 0:29
46cfb3c4b94a470792ace0efdd2df11a,13905223853,113.837765,34.743035, 河南省 , 郑州市,41.9,女,新能源,00后,9,0,,0,,2020-4-12 1:15
878f401c9ca6437585ce1187053c220a,13905223356,113.837765,31.650084, 湖北省 , 随州市,35.6,男,教育和培训,80后,8,1,2020-4-12 1:06,0,,2020-4-12 4:35
44165cf545734bf6a114aa641479e828,15895252169,109.275236,34.255614, 陕西省 , 西安市,30.8,女,020,90后,8,0,,1,15152049060,2020-4-12 5:07
475976fccf3647da9019c4a447e9ae6f,15895257014,108.097809,34.560589, 陕西省 , 咸阳市
```

```
市,83.6,女,制造业,80后,7,0,,0,,2020-4-12 23:03
52dffa4535e846fb8985f87008ae2fc9,13905222462,105.448598,35.167162,甘肃省,定西市,72.9,男,软件工程,90后,1,0,,1,15152046339,2020-4-12 9:39
```

我们可以看到，一行就是一条打车订单数据，而且，一条数据是以逗号来进行分隔的，逗号分隔出来一个个的字段。我们需要了解每个字段的意义，这样，将来我们才能分析出来结果。

2.2 用户打车订单日志

每当用户发起打车时，后台系统都会产生一条日志数据，并形成文件。



```
b05b0034cba34ad4a707b4e67f681c71,15152042581,109.348825,36.068516,陕西省,延安市,78.2,男,软件工程,70后,4,1,2020-4-12 20:54,0,,2020-4-12 20:06
```

这条日志包含了以下这些字段：

orderId	订单id
---------	------

telephone	打车用户手机
long	用户发起打车的经度
lat	用户发起打车的纬度
province	所在省份
city	所在城市
es_money	预估打车费用
gender	用户信息 - 性别
profession	用户信息 - 行业
age_range	年龄段 (70后、80后、...)
tip	小费
subscribe	是否预约 (0 - 非预约、1 - 预约)
sub_time	预约时间
is_agent	是否代叫 (0 - 本人、1 - 代叫)
agent_telephone	预约人手机
order_time	订单时间

2.3 用户取消订单日志



当用户取消订单时，也会在系统后台产生一条日志。用户需求选择取消订单的原因。

ae5c114b4c014634840c24373bcb83bb,13905227376,103.719252,26.314714,云 南 省 , 曲 靖 市,19.5,男,新能源,80后,4,2020-4-12 8:44

日志包含了以下这些字段：

orderId	订单ID
cstm_telephone	客户联系电话
long	取消订单的经度
lat	取消订单的纬度
province	所在省份
city	所在城市
es_distance	预估距离
gender	性别
profession	行业

age_range	年龄段
reason	取消订单原因 (1 - 选择了其他交通方式、2 - 与司机达成一致，取消订单、3 - 投诉司机没来接我、4 - 已不需要用车、5 - 无理由取消订单)
cancel_time	取消时间

2.4 用户支付日志



用户点击确认支付后，系统后台会将用户的支持信息保存为一条日志。

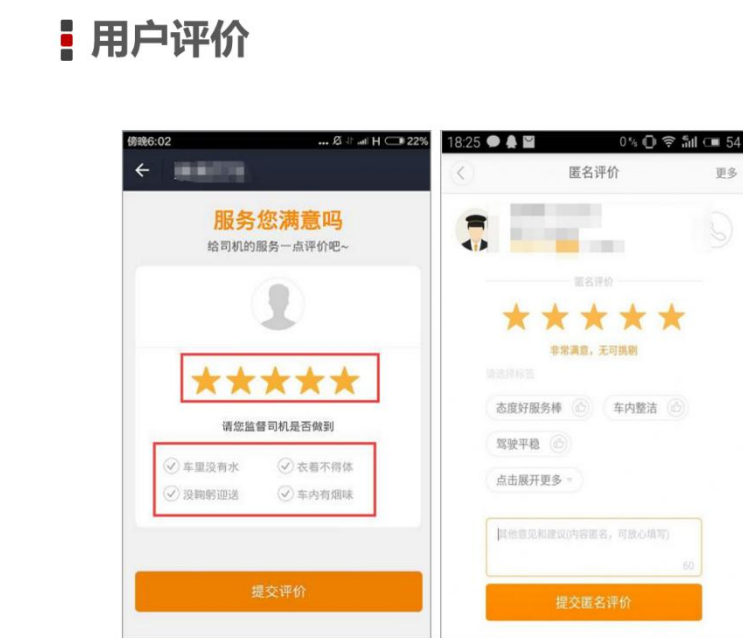
c182c4751bbb412e853e834803801352,b05b0034cba34ad4a707b4e67f681c71,109.348825,36.068516,陕西省,延安市,124.5,72.3,8,44.6,0,,1,9.2,2020-4-12 16:37

日志包含了以下这些字段：

ID	支付订单ID
orderId	订单ID
lng	目的地的经度（支付地址）
lat	目的地的纬度（支付地址）

province	省份
city	城市
total_money	车费总价
real_pay_money	实际支付总额
passenger_additional_money	乘客额外加价
base_money	车费合计
has_coupon	是否使用优惠券（0 - 不使用、1 - 使用）
coupon_total	优惠券合计
pay_way	支付方式（0 - 微信支付、1 - 支付宝支付、2 - QQ钱包支付、3 - 一卡通银行卡支付）
mileage	里程（单位公里）
pay_time	支付时间

2.5 用户评价日志



用户评价

用户不满意就会有这种评价：

备注 枪毙你那一天我一定会去观看，等你埋了再去你坟头蹦迪

但我们点击提交评价后，系统后台也会产生一条日志。


```
f9a212bf7a1f4f7399e2bea018ff52b3,b05b0034cba34ad4a707b4e67f681c71,15152042581,陕西省,延安市,5,2020-4-12 17:34
```

日志包含了以下这些字段：

id	评价日志唯一ID
orderId	订单ID
passenger_telephone	用户电话
passenger_province	用户所在省份
passenger_city	用户所在城市
level	评价等级（1 - 一颗星、... 5 - 五星）
evaluate_time	评价时间

3. 构建数据仓库

3.1 业务分析

我们的目标是分析用户打车的订单，进行各类的指标计算（指标，例如：订单的总数、订单的总支付金额等等）。我们之前学习过了HDFS以及Hive，所以，我们可以将数据上传到HDFS保存下来，每天都可以进行上传，HDFS可以保存海量的数据。同时，我们学习过了Hive，可以将HDFS中的数据文件，对应到Hive的表中。但需要考虑一个问题，就是业务系统的日志数据不一定是能够直接进行分析的，例如：我们需要分析不同时段订单占比，凌晨有多少订单、早上有多少订单、上午有多少订单等。但是，我们发现，原始的日志文件中，并没有区分该订单的是哪个时间段的字段。所以，我们需要对日志文件的原始数据进行预处理，才能进行分析。

我们会有这么几类数据要考虑：

1. 原始日志数据（业务系统中保存的日志文件数据）
2. 预处理后的数据
3. 分析结果数据

这些数据我们都通过Hive来进行处理，因为Hive可以将数据映射为一张张的表，然后就可以通过编写HQL来处理数据了，简单、快捷、高效。为了区分以上这些数据，我们将这些数据对应的表分别

保存在不同的数据库中。

3.2 数仓分层

为了方便组织、管理上述的三类数据，我们将数仓分成不同的层，简单来说，就是分别将三类不同的数据保存在Hive的不同数据库中。

数据类别	数据库名	分层名
原始日志数据（业务系统中保存的日志文件数据）	ods_didi	临时存储层
预处理后的数据	dw_didi	数据仓库层
分析结果数据	app_didi	应用层

3.3 创建数据库

```
-- 1.1 创建ods库
create database if not exists ods_didi;
-- 1.2 创建dw库
create database if not exists dw_didi;
-- 1.3 创建app库
create database if not exists app_didi;
```

4. 创建表

4.1 ods创建用户打车订单表

根据用户打车订单数据，我们需要创建按照日期分区的表：

```
-- 2.1 创建订单表结构
create table if not exists ods_didi.t_user_order(
    orderId string comment '订单id',
    telephone string comment '打车用户手机',
    lng string comment '用户发起打车的经度',
```

```
lat string comment '用户发起打车的纬度',
province string comment '所在省份',
city string comment '所在城市',
es_money double comment '预估打车费用',
gender string comment '用户信息 - 性别',
profession string comment '用户信息 - 行业',
age_range string comment '年龄段（70后、80后、...）',
tip double comment '小费',
subscribe integer comment '是否预约（0 - 非预约、1 - 预约）',
sub_time string comment '预约时间',
is_agent integer comment '是否代叫（0 - 本人、1 - 代叫）',
agent_telephone string comment '预约人手机',
order_time string comment '预约时间'
)
partitioned by (dt string comment '时间分区')
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

4.2 ods创建取消订单表

```
create table if not exists ods_didi.t_user_cancel_order(
    orderId string comment '订单ID',
    cstm_telephone string comment '客户联系电话',
    lng string comment '取消订单的经度',
    lat string comment '取消订单的纬度',
    province string comment '所在省份',
    city string comment '所在城市',
    es_distance double comment '预估距离',
    gender string comment '性别',
    profession string comment '行业',
    age_range string comment '年龄段',
    reason integer comment '取消订单原因（1 - 选择了其他交通方式、2 - 与司机达成一致，取消订单、3 - 投诉司机没来接我、4 - 已不需要用车、5 - 无理由取消订单）',
    cancel_time string comment '取消时间'
)
partitioned by (dt string comment '时间分区')
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

4.3 ods创建订单支付表

```
create table if not exists ods_didi.t_user_pay_order(  
    id string comment '支付订单 ID',  
    orderId string comment '订单 ID',  
    lng string comment '目的地的经度（支付地址）',  
    lat string comment '目的地的纬度（支付地址）',  
    province string comment '省份',  
    city string comment '城市',  
    total_money double comment '车费总价',  
    real_pay_money double comment '实际支付总额',  
    passenger_additional_money double comment '乘客额外加价',  
    base_money double comment '车费合计',  
    has_coupon integer comment '是否使用优惠券（0 - 不使用、1 - 使用）',  
    coupon_total double comment '优惠券合计',  
    pay_way integer comment '支付方式（0 - 微信支付、1 - 支付宝支付、3 - QQ 钱包  
支付、4 - 一网通银行卡支付）',  
    mileage double comment '里程（单位公里）',  
    pay_time string comment '支付时间'  
)  
partitioned by (dt string comment '时间分区')  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

4.4 ods创建用户评价表

```
create table if not exists ods_didi.t_user_evaluate(  
    id string comment '评价日志唯一ID',  
    orderId string comment '订单ID',  
    passenger_telephone string comment '用户电话',  
    passenger_province string comment '用户所在省份',  
    passenger_city string comment '用户所在城市',  
    eva_level integer comment '评价等级（1 - 一颗星、... 5 - 五星）',  
    eva_time string comment '评价时间'  
)  
partitioned by (dt string comment '时间分区')  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

4.5 VSCode列操作细节

1. 使用alt + shift + 拖动鼠标左键进入列编辑
2. 按照 shift + ctrl + 左箭头/右箭头可以选中对应的单词
3. 在列编辑模式可以使用 复制/粘贴 功能
4. 在列标记的时候可以使用 Home/End 键将光标移动到开头或者结尾

5. 添加分区

5.1 基于T+1的日期分区

大规模数据的处理，必须要构建分区。我们此处的需求每天都会进行数据分析，采用的是T+1的模式。就是假设今天是2020-01-01，那么1月1日的分析结果在第二天才能看到，也就是2020-01-02查看到上一天的数据分析结果。此处，我们采用最常用的分区方式，使用日期来进行分区。

5.2 创建分区

```
-- 在订单表中创建 2020-04-12 分区
alter table ods_didi.t_user_order add if not exists partition(dt='2020-04-12');

-- 在取消订单表中创建 2020-04-12 分区
alter table ods_didi.
t_user_cancel_order add if not exists partition(dt='2020-04-12');

-- 在支付表中创建 2020-04-12 分区
alter table ods_didi.t_user_pay_order add if not exists partition(dt='2020-04-12
');

-- 在用户评价订单表创建 2020-04-12 分区
alter table ods_didi.t_user_evaluate add if not exists partition(dt='2020-04-12'
);

-- 查看表分区
show partitions ods_didi.t_user_order;
```

6. 数据上传HDFS

6.1 上传数据到分区对应的HDFS路径

我们已经在Hive中建立好了数据库、表、以及分区。接下来，我们需要将打车的日志数据上传到HDFS分布式文件系统中。然后，我们就可以开始进行数据处理、分析了。

1. 在/root目录下创建 data/didi 文件夹

```
mkdir -p /root/data/didi
```

2. 将数据集文件上传到指定Hadoop分区的文件夹

上传订单数据

```
hadoop fs -put /root/data/didi/order.csv /user/hive/warehouse/ods_didi.db/t_user_order/dt=2020-04-12
```

上传取消订单数据

```
hadoop fs -put /root/data/didi/cancel_order.csv /user/hive/warehouse/ods_didi.db/t_user_cancel_order/dt=2020-04-12
```

上传支付订单数据

```
hadoop fs -put /root/data/didi/pay.csv /user/hive/warehouse/ods_didi.db/t_user_pay_order/dt=2020-04-12
```

上传评价数据

```
hadoop fs -put /root/data/didi/evaluate.csv /user/hive/warehouse/ods_didi.db/t_user_evaluate/dt=2020-04-12
```

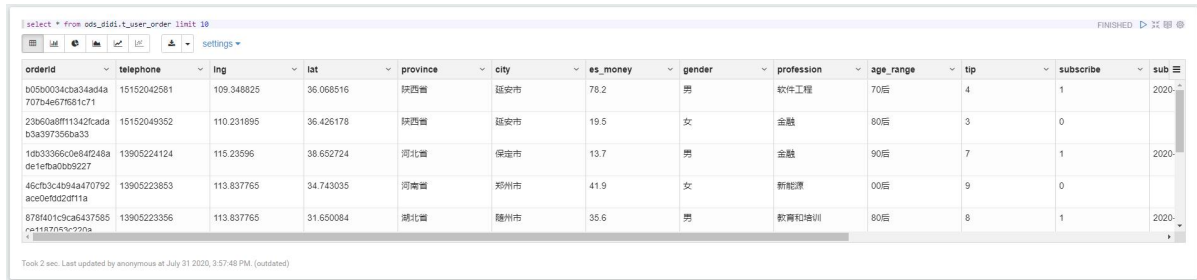
3. 在HDFS中确认文件是否已经上传

```
hadoop fs -ls /user/hive/warehouse/ods_didi.db/t_user_order/dt=2020-04-12
hadoop fs -ls /user/hive/warehouse/ods_didi.db/t_user_cancel_order/dt=2020-04-12
hadoop fs -ls /user/hive/warehouse/ods_didi.db/t_user_pay_order/dt=2020-04-12
hadoop fs -ls /user/hive/warehouse/ods_didi.db/t_user_evaluate/dt=2020-04-12
```

6.2 在Hive中检查数据是否正确映射

在Zeppelin中执行以下一段HQL，检查数据是否都已经正确映射。


```
select * from ods_didi.t_user_order limit 10
```



orderid	telephone	lng	lat	province	city	es_money	gender	profession	age_range	tip	subscribe	sub
b05b0034c3ba34ad4a707b4e67681c71	15152042581	109.348825	36.068516	陕西省	延安市	78.2	男	软件工程	70后	4	1	2020-
23b50a0ff11342fcada03a397356ba35	15152049352	110.231895	36.426178	陕西省	延安市	19.5	女	金融	80后	3	0	
1db33365c0e84f248a4e1efba0b09227	13905224124	115.23596	38.652724	河北省	保定市	13.7	男	金融	90后	7	1	2020-
46c7b3c4b94a70792ace0e9fd2df11a	13905223853	113.837765	34.743035	河南省	郑州市	41.9	女	新能源	00后	9	0	
878f401c9ca6437585ae118705c7229a	13905223356	113.837765	31.650084	湖北省	随州市	35.6	男	教育和培训	80后	8	1	2020-

7. 数据预处理

1. 现在数据已经准备好了，接下来我们需要对ods层中的数据进行预处理。数据预处理是数据仓库开发中的一个重要环节。目的主要是让预处理后的数据更容易进行数据分析，并且能够将一些非法的数据处理掉，避免影响实际的统计结果。

7.1 dw层创建宽表

在进行预处理之前，先要把预处理之后保存数据的表创建出来。它包含以下字段。

字段名	说明
orderId	订单id
telephone	打车用户手机
lng	用户发起打车的经度
lat	用户发起打车的纬度
province	所在省份
city	所在城市
es_money	预估打车费用
gender	用户信息
profession	用户信息
age_range	年龄段（70后、80后、
tip	小费



subscribe	是否预约（0 - 非预约、1 - 预约）
subscribe_name	是否预约名称
sub_time	预约时间
is_agent	是否代叫（0 - 本人、1 - 代叫）
is_agent_name	是否代叫名称
agent_telephone	预约人手机
order_date	预约日期，yyyy-MM-dd
order_year	年
order_month	月
order_day	日
order_hour	小时
order_time_range	时间段
order_time	预约时间

建表语句如下：

```
create table if not exists dw_didi.t_user_order_wide(  
    orderId string comment '订单id',  
    telephone string comment '打车用户手机',  
    lng string comment '用户发起打车的经度',  
    lat string comment '用户发起打车的纬度',  
    province string comment '所在省份',  
    city string comment '所在城市',  
    es_money double comment '预估打车费用',  
    gender string comment '用户信息 - 性别',  
    profession string comment '用户信息 - 行业',  
    age_range string comment '年龄段（70后、80后、...）',  
    tip double comment '小费',  
    subscribe integer comment '是否预约（0 - 非预约、1 - 预约）',
```

```
subscribe_name string comment '是否预约名称',
sub_time string comment '预约时间',
is_agent integer comment '是否代叫（0 - 本人、1 - 代叫）',
is_agent_name string comment '是否代叫名称',
agent_telephone string comment '预约人手机',
order_date string comment '预约时间，yyyy-MM-dd',
order_year string comment '年',
order_month string comment '月',
order_day string comment '日',
order_hour string comment '小时',
order_time_range string comment '时间段',
order_time string comment '预约时间'
)
partitioned by (dt string comment '时间分区')
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

7.2 用户订单处理

此处，我们需要在预处理之前考虑以下需求：

1. 过滤掉order_time长度小于8的数据，如果小于8，表示这条数据不合法，不应该参加统计。
2. 将一些0、1表示的字段，处理为更容易理解的字段。例如：subscribe字段，0表示非预约、1表示预约。我们需要添加一个额外的字段，用来展示非预约和预约，这样将来我们分析的时候，跟容易看懂数据。
3. order_time字段为2020-4-12 1:15，为了将来更方便处理，我们统一使用类似 2020-04-12 01:15来表示，这样所有的order_time字段长度是一样的。并且将日期获取出来
4. 为了方便将来按照年、月、日、小时统计，我们需要新增这几个字段。
5. 后续要分析一天内，不同时段的订单量，我们需要在预处理过程中将订单对应的时间段提前计算出来。例如：1:00-5:00为凌晨。对应关系如下：

1: 00 – 5:00	凌晨
5:00 – 8:00	早上
8:00 – 11:00	上午
11:00 – 13:00	中午



13:00 – 17:00	下午
17:00 – 19:00	晚上
19:00 – 20:00	半夜
20:00 – 24:00	深夜
0:00 – 1:00	凌晨

我们按照上面的需求，一步一步将预处理的SQL代码编写出来

```
select
    orderId,
    telephone,
    lng,
    lat,
    province,
    city,
    es_money,
    gender,
    profession,
    age_range,
    tip,
    subscribe,
    case when subscribe = 0 then '非预约'
         when subscribe = 1 then '预约'
    end as subscribe_name,
    sub_time,
    is_agent,
    case when is_agent = 0 then '本人'
         when is_agent = 1 then '代叫'
    end as is_agent_name,
    agent_telephone,
    date_format(order_time, 'yyyy-MM-dd') as order_date,
    year(date_format(order_time, 'yyyy-MM-dd')) as order_year,
    month(date_format(order_time, 'yyyy-MM-dd')) as order_month,
    day(date_format(order_time, 'yyyy-MM-dd')) as order_day,
    hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) as order_hour,
    case when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 1 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 5 then '凌晨'
         when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 5 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 8 then '早上'
         when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 8 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 11 then '上午'
         when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 11 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 13 then '中午'
         when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 13 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 17 then '下午'
         when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 17 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 19 then '晚上'
```



```
        when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 19 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 20 then '半夜'
        when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 20 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 24 then '深夜'
        when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) >= 0 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 1 then '深夜'
        else 'N/A'
    end as order_time_range,
    date_format(order_time, 'yyyy-MM-dd HH:mm') as order_time
from ods_didi.t_user_order where dt = '2020-04-12' and length(order_time) > 8
;
```

7.3 将数据加载到dw层宽表

HQL编写好后，为了方便后续分析，我们需要将预处理好的数据写入到之前创建的宽表中。注意：宽表也是一个分区表，所以，写入的时候一定要指定对应的分区。

```
insert overwrite table dw_didi.t_user_order_wide partition(dt='2020-04-12')
select
    orderId,
    telephone,
    long,
    lat,
    province,
    city,
    es_money,
    gender,
    profession,
    age_range,
    tip,
    subscribe,
    case when subscribe = 0 then '非预约'
        when subscribe = 1 then '预约'
    end as subscribe_name,
    sub_time,
    is_agent,
    case when is_agent = 0 then '本人'
        when is_agent = 1 then '代叫'
    end as is_agent_name,
    agent_telephone,
    date_format(order_time, 'yyyy-MM-dd') as order_date,
    year(date_format(order_time, 'yyyy-MM-dd')) as order_year,
    month(date_format(order_time, 'yyyy-MM-dd')) as order_month,
    day(date_format(order_time, 'yyyy-MM-dd')) as order_day,
    hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) as order_hour,
```

```
case when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 1 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 5 then '凌晨'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 5 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 8 then '早上'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 8 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 11 then '上午'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 11 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 13 then '中午'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 13 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 17 then '下午'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 17 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 19 then '晚上'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 19 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 20 then '半夜'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) > 20 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 24 then '深夜'
      when hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) >= 0 and hour(date_format(order_time, 'yyyy-MM-dd HH:mm')) <= 1 then '深夜'
      else 'N/A'
end as order_time_range,
date_format(order_time, 'yyyy-MM-dd HH:mm') as order_time
from ods_didi.t_user_order where dt = '2020-04-12' and length(order_time) > 8
;
```

8. 订单分析

数据处理好后，我们就可以开始分析了。

8.1 总订单笔数分析

8.1.1 编写HQL分析

计算4月12日总订单笔数

```
-- 1. 计算4月12日总订单笔数
select
    count(orderid) as total_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
;
```


8.1.2 app层建表

数据分析好了，但要知道，我们处理大规模数据，每次处理都需要占用较长时间，所以，我们可以将计算好的数据，直接保存下来。将来，我们就可以快速查询数据结果了。所以，我们可以提前在app层创建好表。此处，我们要保存的数据为某天的总订单数。我们要保存以下几个字段：

1. 时间（哪天的订单总数）
2. 订单总数

```
-- 创建保存日期对应订单笔数的app表
create table if not exists app_didi.t_order_total(
    date string comment '日期（年月日）',
    count integer comment '订单笔数'
)
partitioned by (month string comment '年月，yyyy-MM')
row format delimited fields terminated by ','
;
```

8.1.3 加载数据到app表

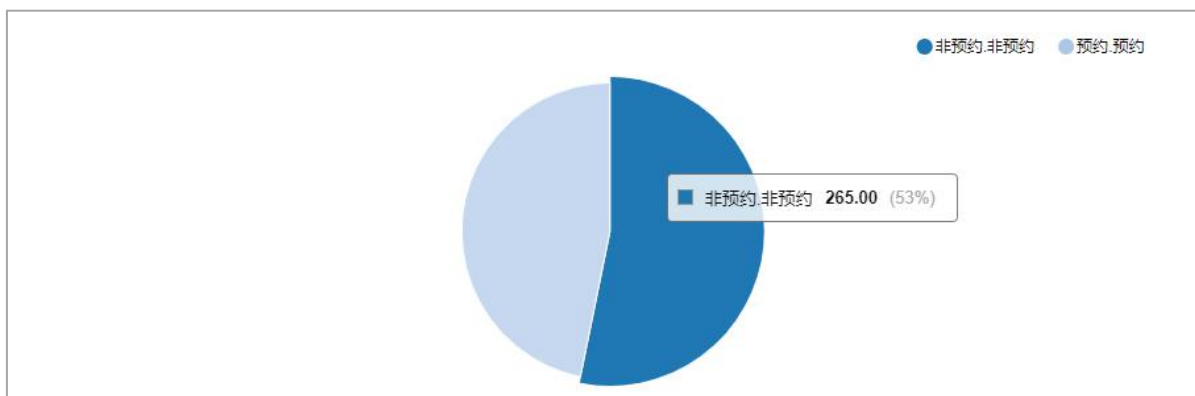
此处因为结果的数据不会很多，所以，我们只需要基于年月来分区就可以了。

```
insert overwrite table app_didi.t_order_total partition(month='2020-04')
select
    '2020-04-12',
    count(orderid) as total_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
;
```

8.1.4 使用Zeppelin展示数据

date	count	month
2020-04-12	498	2020-04

8.2 预约订单/非预约订单占比分析



8.2.1 编写HQL分析

```
select
    subscribe_name,
    count(*) as order_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
group by
    subscribe_name
;
```

8.2.2 app层建表

包含以下几个字段：

1. 日期
2. 是否预约
3. 订单数量

建表语句：

```
create table if not exists app_didi.t_order_subscribe_total(  
    date string comment '日期',  
    subscribe_name string comment '是否预约',  
    count integer comment '订单数量'  
)  
partitioned by (month string comment '年月, yyyy-MM')  
row format delimited fields terminated by ','  
;
```

8.2.3 加载数据到app表

```
insert overwrite table app_didi.t_order_subscribe_total partition(month = '2020-  
04')  
select  
    '2020-04-12',  
    subscribe_name,  
    count(*) as order_cnt  
from  
    dw_didi.t_user_order_wide  
where  
    dt = '2020-04-12'  
group by  
    subscribe_name  
;
```

8.2.4 使用Zeppelin展示数据

1. 在Zeppelin中创建一个新的note
2. 在note中设置宽度、并查询出来指定某天的总订单数，并展示出来

```
select
```

```
*  
from  
    app_didi.t_order_subscribe_total  
where  
    date = '2020-04-12'  
;
```

8.3 不同时段订单占比分析

8.3.1 编写HQL分析

```
select  
    order_time_range,  
    count(*) as order_cnt  
from  
    dw_didi.t_user_order_wide  
where  
    dt = '2020-04-12'  
group by  
    order_time_range  
;
```

8.3.2 app层建表

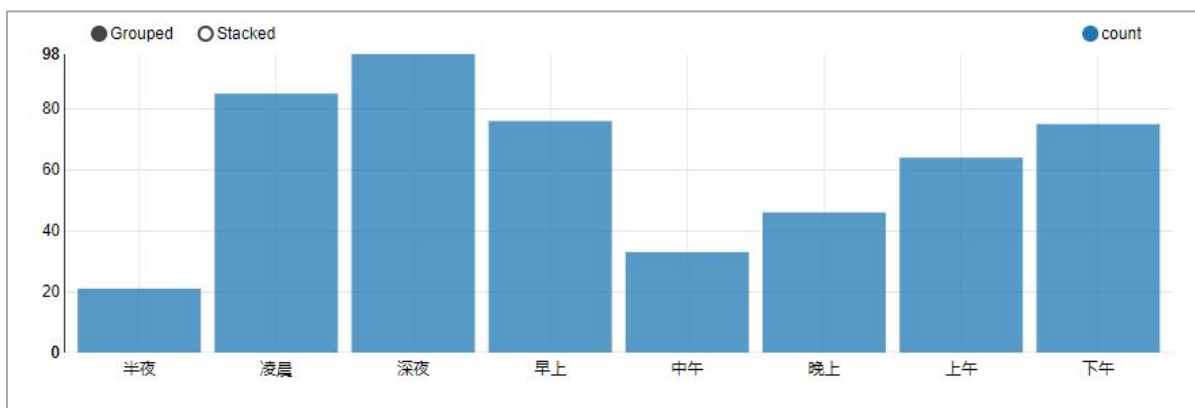
```
create table if not exists app_didi.t_order_timerange_total(  
    date string comment '日期',  
    timerange string comment '时间段',  
    count integer comment '订单数量'  
)  
partitioned by (month string comment '年月, yyyy-MM')  
row format delimited fields terminated by ','  
;
```

8.3.3 加载数据到app表

```
insert overwrite table app_didi.t_order_timerange_total partition(month = '2020-04')
```

```
select
    '2020-04-12',
    order_time_range,
    count(*) as order_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
group by
    order_time_range
;
```

8.3.4 使用Zeppelin展示数据



查询HQL语句：

```
select
    *
from
    app_didi.t_order_timerange_total
where
    date = '2020-04-12'
;
```

配置Zeppelin：



8.4 不同地域订单占比分析（省份）

8.4.1 编写HQL分析

```
select
    province,
    count(*) as order_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
group by
    province
;
```

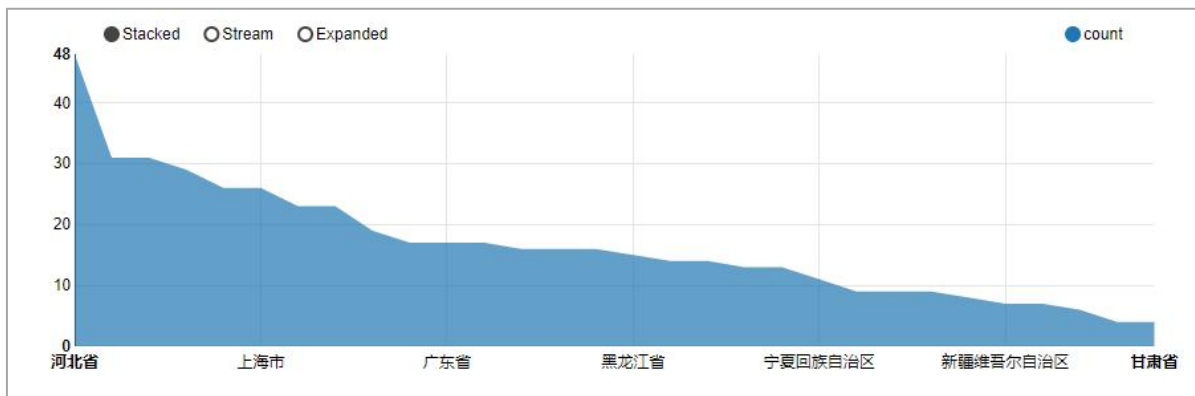
8.4.2 app层建表

```
create table if not exists app_didi.t_order_province_total(
    date string comment '日期',
    province string comment '省份',
    count integer comment '订单数量'
)
partitioned by (month string comment '年月, yyyy-MM')
row format delimited fields terminated by ','
;
```

8.4.3 加载数据到app表


```
insert overwrite table app_didi.t_order_province_total partition(month = '2020-04-12')
select
    '2020-04-12',
    province,
    count(*) as order_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
group by
    province
;
```

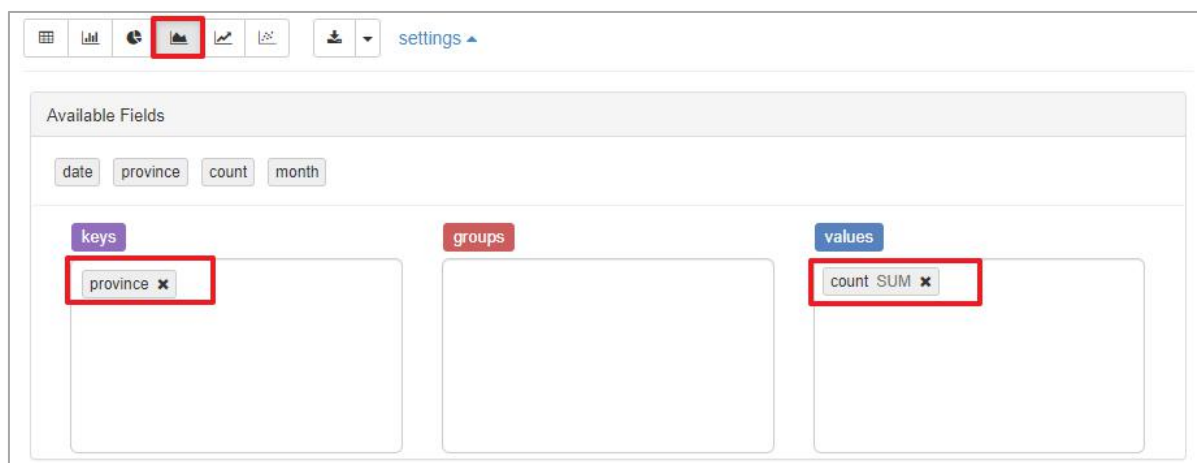
8.4.4 使用Zeppelin展示数据



查询HQL语句：

```
select
    *
from
    app_didi.t_order_province_total
where
    date = '2020-04-12'
;
```

配置Zeppelin：



8.5 不同年龄段订单占比分析

8.5.1 编写HQL分析

```
select
    age_range,
    count(*) as order_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
group by
    age_range
;
```

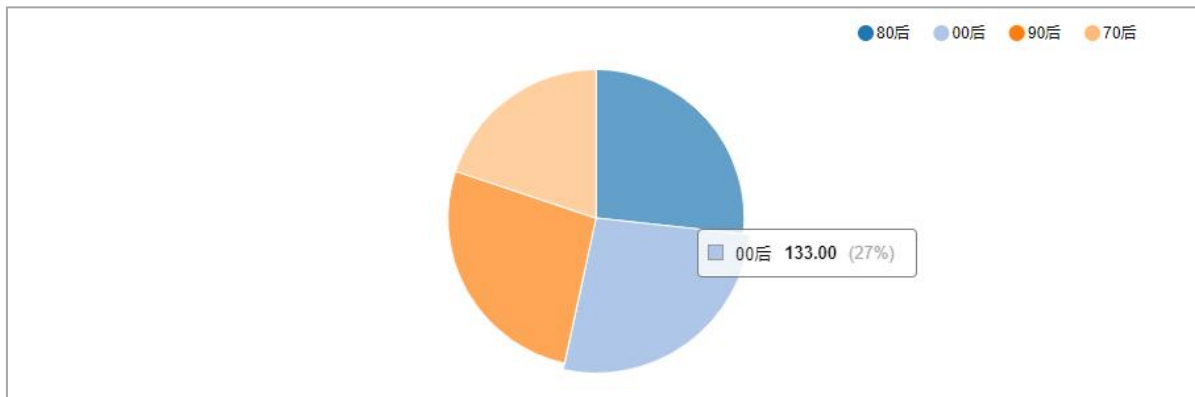
8.5.2 app层建表

```
create table if not exists app_didi.t_order_agerange_total(
    date string comment '日期',
    age_range string comment '年龄段',
    count integer comment '订单数量'
)
partitioned by (month string comment '年月, yyyy-MM')
row format delimited fields terminated by ','
;
```

8.5.3 加载数据到app表

```
insert overwrite table app_didi.t_order_agerange_total partition(month = '2020-04')
select
    '2020-04-12',
    age_range,
    count(*) as order_cnt
from
    dw_didi.t_user_order_wide
where
    dt = '2020-04-12'
group by
    age_range
;
```

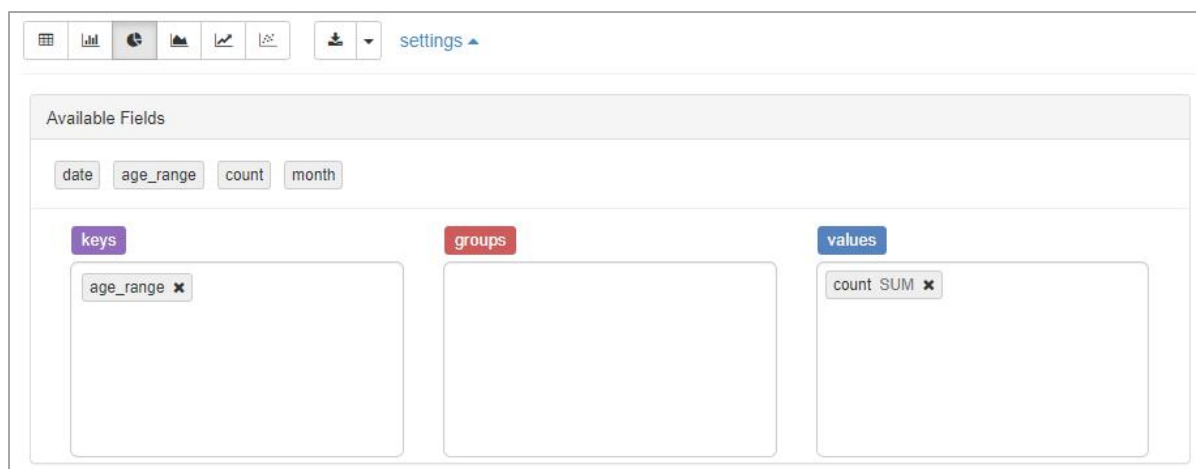
8.5.4 使用Zeppelin展示数据



查询HQL语句：

```
select
    *
from
    app_didi.t_order_agerange_total
where
    date = '2020-04-12'
;
```

配置Zeppelin：



9. 作业

9.1 取消订单主题

开发步骤：

1. 创建dw层表，用于保存ods层的数据拉宽后的数据
2. 编写HQL代码，对ods层表的数据进行拉宽
3. 创建app层表，用于保存计算后的结果
4. 编写分析的HQL代码，并将数据加载到app表
5. 使用Zeppelin将数据展示出来

9.1.1 取消订单笔数

统计2020-04-12这一天取消的订单总计

9.1.2 取消订单原因分析

统计2020-04-12这一天订单取消的原因

9.1.3 支付主题

9.1.3.1 总支付金额/总优惠券金额/用户打赏总金额

统计2020-04-12这一天的总支付金额、总优惠券使用基恩、总用户打赏总金额

9.1.3.2 不同支付方式订单总额分析

统计2020-04-12这一天不同方式的订单总额

9.1.3.3 不同区域支付总额分析

统计2020-04-12不同省份的订单总额

参考代码参见资料中的「作业代码」