Day13_Hive & 函数

大数据-张军锋 Day13 Hive SQL 函数

java代码操作hive

创建maven工程，导入依赖文件

```
<dependencies>
    <dependency>
        <groupId>org.apache.hive</groupId>
        <artifactId>hive-jdbc</artifactId>
        <version>2.3.0</version>
    </dependency>
    <dependency>
        <groupId>jdk.tools</groupId>
        <artifactId>jdk.tools</artifactId>
        <version>1.8</version>
        <scope>system</scope>
        <systemPath>${JAVA_HOME}/lib/tools.jar</systemPath>
    </dependency>
</dependencies>
```

编写连接小工具

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class HiveJdbcUtils {

    public static final String DRIVER_CLASS = "org.apache.hive.jdb
c.HiveDriver";
    public static final String URL = "jdbc:hive2://master:10000/db1
4";
    public static final String USERNAME = "root";
    public static final String PASSWORD = "";
    private static Connection connection;
    /**
    * getConnection 获取连接
    * @param @return 参数
    * @return Connection 返回类型
    * @Exception 异常对象
    */
    public static Connection getConnection(){
        try {
            Class.forName(DRIVER_CLASS);
            connection = DriverManager.getConnection(URL,USERNAME,P
ASSWORD);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return connection;
    }

    /**
    * close 关闭连接
    * @param @param connection 参数
    * @return void 返回类型
    * @Exception 异常对象
    */
    public static void close(Connection connection) {
        try {
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

hive的基本操作

创建表

```
public static void createTable() throws SQLException {
    Connection connection = HiveJdbcUtils.getConnection();
    Statement statement = connection.createStatement();
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("create table table_form_java(");
    stringBuilder.append("test1 string");
    stringBuilder.append(",test2 int");
    stringBuilder.append(",test3 string");
    stringBuilder.append(")stored as textfile");
    statement.execute(stringBuilder.toString());
    ResultSet result = statement.executeQuery("show tables");
    while(result.next()){
        System.out.println(result.getString(1));
    }
}
```

Hive 函数

hive 中内置了很多函数，具体使用用法详见官方文档
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

查看所有函数 `show functions`
查看函数的具体用法 `describe function last_day` 或者 `describe function extended last_day`

返回当前月的最后一天

```
-- Returns the last day of the month
select last_day(to_date('2017-10-01'))
select last_day('2017-10-01')
```

字符串拼接

```
show tables;
select '姓名:' || emp_name || '\t薪水:'|| salary
from dw_employee

select concat(emp_name,':',salary)
from dw_employee

describe function extended concat
```

hive支持正则表达式

```
describe function extended regexp
select * from dw_employee
where status regexp '(.{4})'
```

复杂类型构造方法

```
-- 复杂类型构造方法
select map(emp_name,status)
from dw_employee
```

关系运算

> **等值比较：** =
> **语法：**A=B
> **操作类型：**所有基本类型
> **描述：** 如果表达式A与表达式B相等，则为TRUE；否则为FALSE

```
select 1 from iteblog where 1=1;
1
```

> **不等值比较：**<>
> **语法：**A <> B
> **操作类型：**所有基本类型
> **描述:：**如果表达式A为NULL，或者表达式B为NULL，返回NULL；如果表达式A与表达式B不相等，则为TRUE；否则为FALSE

```
select 1 from iteblog where 1 <> 2;
1
```

小于比较:：<
语法：A < B
操作类型：所有基本类型
描述:：如果表达式A为NULL，或者表达式B为NULL，返回NULL；如果表达式A小于表达式B，则为TRUE；否则为FALSE

```
select 1 from iteblog where 1 < 2;
1
```

小于等于比较： <=
语法:：A <= B
操作类型:：所有基本类型
描述:：如果表达式A为NULL，或者表达式B为NULL，返回NULL；如果表达式A小于或者等于表达式B，则为TRUE；否则为FALSE

```
select 1 from iteblog where 1 < = 1;
1
```

大于比较：>
语法：A > B
操作类型：所有基本类型
描述：如果表达式A为NULL，或者表达式B为NULL，返回NULL；如果表达式A大于表达式B，则为TRUE；否则为FALSE

```
select 1 from iteblog where 2 > 1;
1
```

大于等于比较：>=
语法：A >= B
操作类型：所有基本类型
描述：如果表达式A为NULL，或者表达式B为NULL，返回NULL；如果表达式A大于或者等于表达式B，则为TRUE；否则为FALSE

```
select 1 from iteblog where 1 >= 1;
1
```

注意：String的比较要注意（用的时间比较可以先 to_date 之后再比较）

```
select * from iteblog;
OK
2011111209 00:00:00     2011111209
```

```
select a, b, a<b, a>b, a=b from iteblog;
2011111209 00:00:00     2011111209      false   true    false
```

空值判断：IS NULL
语法： A IS NULL
操作类型：所有类型
描述：如果表达式A的值为NULL，则为TRUE；否则为FALSE

```
select 1 from iteblog where null is null;
1
```

非空判断：IS NOT NULL
语法：A IS NOT NULL
操作类型：所有类型
描述：如果表达式A的值为NULL，则为FALSE；否则为TRUE

```
select 1 from iteblog where 1 is not null;
1
```

LIKE比较：LIKE
语法：A LIKE B
操作类型： strings
描述： 如果字符串A或者字符串B为NULL，则返回NULL；如果字符串A符合表达式B 的正则语法，则为TRUE；否则为FALSE。B中字符”_”表示任意单个字符，而字符”%”表示任意数量的字符。

```
select 1 from iteblog where 'football' like 'foot%';
1
select 1 from iteblog where 'football' like 'foot____';
1
```

注意：否定比较时候用NOT A LIKE B

```
select 1 from iteblog where NOT 'football' like 'fff%';
1
```

> **JAVA的LIKE操作:** RLIKE
> **语法:** A RLIKE B
> **操作类型:** strings
> **描述:** 如果字符串A或者字符串B为NULL，则返回NULL；如果字符串A符合JAVA正则表达式B的正则语法，则为TRUE；否则为FALSE。

```
select 1 from iteblog where 'footbar' rlike '^f.*r$';
1
```

注意：判断一个字符串是否全为数字：

```
select 1 from iteblog where '123456' rlike '^\\d+$';
1
select 1 from iteblog where '123456aa' rlike '^\\d+$';
```

> **REGEXP操作:** REGEXP
> **语法:** A REGEXP B
> **操作类型:** strings
> **描述:** 功能与RLIKE相同

```
select 1 from iteblog where 'footbar' REGEXP '^f.*r$';
1
```

数学运算

加法操作: +
语法: A + B
操作类型：所有数值类型
说明：返回A与B相加的结果。结果的数值类型等于A的类型和B的类型的最小父类型（详见数据类型的继承关系）。比如，int + int 一般结果为int类型，而 int + double 一般结果为double类型

```
select 1 + 9 from iteblog;
10
create table iteblog as select 1 + 1.2 from iteblog;
describe iteblog;
_c0     double
```

减法操作: -
语法: A – B
操作类型：所有数值类型
说明：返回A与B相减的结果。结果的数值类型等于A的类型和B的类型的最小父类型（详见数据类型的继承关系）。比如，int – int 一般结果为int类型，而 int – double 一般结果为double类型

```
select 10 - 5 from iteblog;
5
create table iteblog as select 5.6 - 4 from iteblog;
describe iteblog;
_c0     double
```

乘法操作: *
语法: A * B
操作类型：所有数值类型
说明：返回A与B相乘的结果。结果的数值类型等于A的类型和B的类型的最小父类型（详见数据类型的继承关系）。注意，如果A乘以B的结果超过默认结果类型的数值范围，则需要通过cast将结果转换成范围更大的数值类型

```
select 40 * 5 from iteblog;
200
```

除法操作: /
语法: A / B
操作类型：所有数值类型
说明：返回A除以B的结果。结果的数值类型为double

```
select 40 / 5 from iteblog;
8.0
```

注意：hive中最高精度的数据类型是double,只精确到小数点后16位，在做除法运算的时候要特别注意

```
select ceil(28.0/6.99999999999999999999) from iteblog limit 1;
结果为4
select ceil(28.0/6.99999999999999) from iteblog limit 1;
结果为5
```

取余操作: %
语法: A % B
操作类型：所有数值类型
说明：返回A除以B的余数。结果的数值类型等于A的类型和B的类型的最小父类型（详见数据类型的继承关系）。

```
select 41 % 5 from iteblog;
1
select 8.4 % 4 from iteblog;
0.40000000000000036
```

注意：精度在hive中是个很大的问题，类似这样的操作最好通过round指定精度

```
select round(8.4 % 4 , 2) from iteblog;
0.4
```

位与操作: &
语法: A & B
操作类型：所有数值类型
说明：返回A和B按位进行与操作的结果。结果的数值类型等于A的类型和B的类型的最小父类型（详见数据类型的继承关系）。

```
select 4 & 8 from iteblog;
0
select 6 & 4 from iteblog;
4
```

位或操作: |
语法: A | B
操作类型：所有数值类型
说明：返回A和B按位进行或操作的结果。结果的数值类型等于A的类型和B的类型的最小父类型（详见数据类型的继承关系）。

```
select 4 | 8 from iteblog;
12
select 6 | 8 from iteblog;
14
```

位异或操作: ^
语法: A ^ B
操作类型：所有数值类型
说明：返回A和B按位进行异或操作的结果。结果的数值类型等于A的类型和B的类型的最小父类型（详见数据类型的继承关系）。

```
select 4 ^ 8 from iteblog;
12
select 6 ^ 4 from iteblog;
2
```

位取反操作: ~
语法: ~A
操作类型：所有数值类型
说明：返回A按位取反操作的结果。结果的数值类型等于A的类型。

```
select ~6 from iteblog;
-7
select ~4 from iteblog;
-5
```

逻辑运算

> **逻辑与操作:** AND
> **语法:** A AND B
> **操作类型：**boolean
> **说明：**如果A和B均为TRUE，则为TRUE；否则为FALSE。如果A为NULL或B为NULL，则为NULL

```
select 1 from iteblog where 1=1 and 2=2;
1
```

> **逻辑或操作:** OR
> **语法:** A OR B
> **操作类型：**boolean
> **说明：**如果A为TRUE，或者B为TRUE，或者A和B均为TRUE，则为TRUE；否则为FALSE

```
select 1 from iteblog where 1=2 or 2=2;
1
```

> **逻辑非操作:** NOT
> **语法:** NOT A
> **操作类型：**boolean
> **说明：**如果A为FALSE，或者A为NULL，则为TRUE；否则为FALSE

```
select 1 from iteblog where not 1=2;
1
```

数值计算

小数的数据类型 **double float decimal numberic**，一般我们使用**decimal**

取整函数: round
语法: round(double a)
返回值: BIGINT
说明: 返回double类型的整数值部分 （ 遵循四舍五入 ）

```
select round(3.1415926) from iteblog;
3
select round(3.5) from iteblog;
4
create table iteblog as select round(9542.158) from iteblog;
describe iteblog;
_c0     bigint
```

指定精度取整函数: round
语法: round(double a, int d)
返回值: DOUBLE
说明: 返回指定精度d的double类型

```
select round(3.1415926,4) from iteblog;
3.1416
```

向下取整函数: floor
语法: floor(double a)
返回值: BIGINT
说明: 返回等于或者小于该double变量的最大的整数

```
select floor(3.1415926) from iteblog;
3
select floor(25) from iteblog;
25
```

向上取整函数: ceil
语法: ceil(double a)
返回值: BIGINT
说明: 返回等于或者大于该double变量的最小的整数

```
select ceil(3.1415926) from iteblog;
4
select ceil(46) from iteblog;
46
```

向上取整函数: ceiling
语法: ceiling(double a)
返回值: BIGINT
说明: 与ceil功能相同

```
select ceiling(3.1415926) from iteblog;
4
select ceiling(46) from iteblog;
46
```

取随机数函数: rand
语法: rand(),rand(int seed)
返回值: double
说明: 返回一个0到1范围内的随机数。如果指定种子seed，则会等到一个稳定的随机数序列

```
select rand() from iteblog;
0.5577432776034763
select rand() from iteblog;
0.6638336467363424
select rand(100) from iteblog;
0.7220096548596434
select rand(100) from iteblog;
0.7220096548596434
```

自然指数函数: exp
语法: exp(double a)
返回值: double
说明: 返回自然对数e的a次方

```
select exp(2) from iteblog;
7.38905609893065
```

自然对数函数: ln
语法: ln(double a)
返回值: double
说明: 返回a的自然对数

```
select ln(7.38905609893065) from iteblog;
2.0
```

以10为底对数函数: log10
语法: log10(double a)
返回值: double
说明: 返回以10为底的a的对数

```
select log10(100) from iteblog;
2.0
```

以2为底对数函数: log2
语法: log2(double a)
返回值: double
说明: 返回以2为底的a的对数

```
select log2(8) from iteblog;
3.0
```

对数函数: log
语法: log(double base, double a)
返回值: double
说明: 返回以base为底的a的对数

```
select log(4,256) from iteblog;
4.0
```

幂运算函数: pow
语法: pow(double a, double p)
返回值: double
说明: 返回a的p次幂

```
select pow(2,4) from iteblog;
16.0
```

幂运算函数: power
语法: power(double a, double p)
返回值: double
说明: 返回a的p次幂,与pow功能相同

```
select power(2,4) from iteblog;
16.0
```

开平方函数: sqrt
语法: sqrt(double a)
返回值: double
说明: 返回a的平方根

```
select sqrt(16) from iteblog;
4.0
```

二进制函数: bin
语法: bin(BIGINT a)
返回值: string
说明: 返回a的二进制代码表示

```
select bin(7) from iteblog;
111
```

十六进制函数: hex
语法: hex(BIGINT a)
返回值: string
说明: 如果变量是int类型，那么返回a的十六进制表示；如果变量是string类型，则返回该字符串的十六进制表示

```
select hex(17) from iteblog;
11
select hex('abc') from iteblog;
616263
```

反转十六进制函数: unhex
语法: unhex(string a)
返回值: string
说明: 返回该十六进制字符串所代码的字符串

```
select unhex('616263') from iteblog;
abc
select unhex('11') from iteblog;
-
select unhex(616263) from iteblog;
abc
```

进制转换函数: conv
语法: conv(BIGINT num, int from_base, int to_base)
返回值: string
说明: 将数值num从from_base进制转化到to_base进制

```
select conv(17,10,16) from iteblog;
11
select conv(17,10,2) from iteblog;
10001
```

绝对值函数: abs
语法: abs(double a) abs(int a)
返回值: double int
说明: 返回数值a的绝对值

```
select abs(-3.9) from iteblog;
3.9
select abs(10.9) from iteblog;
10.9
```

正取余函数: pmod
语法: pmod(int a, int b),pmod(double a, double b)
返回值: int double
说明: 返回正的a除以b的余数

```
select pmod(9,4) from iteblog;
1
select pmod(-9,4) from iteblog;
3
```

正弦函数: sin
语法: sin(double a)
返回值: double
说明: 返回a的正弦值

```
select sin(0.8) from iteblog;
0.7173560908995228
```

反正弦函数: asin
语法: asin(double a)
返回值: double
说明: 返回a的反正弦值

```
select asin(0.7173560908995228) from iteblog;
0.8
```

余弦函数: cos
语法: cos(double a)
返回值: double
说明: 返回a的余弦值

```
select cos(0.9) from iteblog;
0.6216099682706644
```

反余弦函数: acos
语法: acos(double a)
返回值: double
说明: 返回a的反余弦值

```
select acos(0.6216099682706644) from iteblog;
0.9
```

positive函数: positive
语法: positive(int a), positive(double a)
返回值: int double
说明: 返回a

```
select positive(-10) from iteblog;
-10
select positive(12) from iteblog;
12
```

negative函数: negative
语法: negative(int a), negative(double a)
返回值: int double
说明: 返回-a

```
select negative(-5) from iteblog;
5
select negative(8) from iteblog;
-8
```

日期函数

UNIX时间戳转日期函数: from_unixtime
语法: from_unixtime(bigint unixtime[, string format])
返回值: string
说明: 转化UNIX时间戳（从1970-01-01 00:00:00 UTC到指定时间的秒数）到当前时区的时间格式

```
select from_unixtime(1323308943,'yyyyMMdd') from iteblog;
20111208
```

获取当前UNIX时间戳函数: unix_timestamp
语法: unix_timestamp()
返回值: bigint
说明: 获得当前时区的UNIX时间戳

```
select unix_timestamp() from iteblog;
1323309615
```

日期转UNIX时间戳函数: unix_timestamp
语法: unix_timestamp(string date)
返回值: bigint
说明: 转换格式为”yyyy-MM-dd HH:mm:ss”的日期到UNIX时间戳。如果转化失败，则返回0。

```
select unix_timestamp('2011-12-07 13:01:03') from iteblog;
1323234063
```

指定格式日期转UNIX时间戳函数: unix_timestamp
语法: unix_timestamp(string date, string pattern)
返回值: bigint
说明: 转换pattern格式的日期到UNIX时间戳。如果转化失败，则返回0。

```
select unix_timestamp('20111207 13:01:03','yyyyMMdd HH:mm:ss') from
iteblog;
1323234063
```

日期时间转日期函数: to_date
语法: to_date(string timestamp)
返回值: string
说明: 返回日期时间字段中的日期部分。

```
select to_date('2011-12-08 10:03:01') from iteblog;
2011-12-08
```

日期转年函数: year
语法: year(string date)
返回值: int
说明: 返回日期中的年。

```
select year('2011-12-08 10:03:01') from iteblog;
2011
select year('2012-12-08') from iteblog;
2012
```

日期转月函数: month
语法: month (string date)
返回值: int
说明: 返回日期中的月份。

```
select month('2011-12-08 10:03:01') from iteblog;
12
select month('2011-08-08') from iteblog;
8
```

日期转天函数: day
语法: day (string date)
返回值: int
说明: 返回日期中的天。

```
select day('2011-12-08 10:03:01') from iteblog;
8
select day('2011-12-24') from iteblog;
24
```

日期转小时函数: hour
语法: hour (string date)
返回值: int
说明: 返回日期中的小时。

```
select hour('2011-12-08 10:03:01') from iteblog;
10
```

日期转分钟函数: minute
语法: minute (string date)
返回值: int
说明: 返回日期中的分钟。

```
select minute('2011-12-08 10:03:01') from iteblog;
3
```

日期转秒函数: second
语法: second (string date)
返回值: int
说明: 返回日期中的秒。

```
select second('2011-12-08 10:03:01') from iteblog;
1
```

日期转周函数: weekofyear
语法: weekofyear (string date)
返回值: int
说明: 返回日期在当前的周数。

```
select weekofyear('2011-12-08 10:03:01') from iteblog;
49
```

日期比较函数: datediff
语法: datediff(string enddate, string startdate)
返回值: int
说明: 返回结束日期减去开始日期的天数。

```
select datediff('2012-12-08','2012-05-09') from iteblog;
213
```

日期增加函数: date_add
语法: date_add(string startdate, int days)
返回值: string
说明: 返回开始日期startdate增加days天后的日期。

```
select date_add('2012-12-08',10) from iteblog;
2012-12-18
```

日期减少函数: date_sub
语法: date_sub (string startdate, int days)
返回值: string
说明: 返回开始日期startdate减少days天后的日期。

```
select date_sub('2012-12-08',10) from iteblog;
2012-11-28
```

条件函数

If函数: if
语法: if(boolean testCondition, T valueTrue, T valueFalseOrNull)
返回值: T
说明: 当条件testCondition为TRUE时，返回valueTrue；否则返回valueFalseOrNull

```
select if(1=2,100,200) from iteblog;
200
select if(1=1,100,200) from iteblog;
100
```

非空查找函数: COALESCE
语法: COALESCE(T v1, T v2, …)
返回值: T
说明: 返回参数中的第一个非空值；如果所有值都为NULL，那么返回NULL

```
select COALESCE(null,'100','50') from iteblog;
100
```

条件判断函数： CASE
语法: CASE a WHEN b THEN c [WHEN d THEN e]* [ELSE f] END
返回值: T
说明：如果a等于b，那么返回c；如果a等于d，那么返回e；否则返回f

```
Select case 100 when 50 then 'tom' when 100 then 'mary' else 'tim'
end from iteblog;
mary
Select case 200 when 50 then 'tom' when 100 then 'mary' else 'tim'
end from iteblog;
tim
```

条件判断函数： CASE
语法: CASE WHEN a THEN b [WHEN c THEN d]* [ELSE e] END
返回值: T
说明：如果a为TRUE,则返回b；如果c为TRUE，则返回d；否则返回e

```
select case when 1=2 then 'tom' when 2=2 then 'mary' else 'tim' end
from iteblog;
mary
select case when 1=1 then 'tom' when 2=2 then 'mary' else 'tim' end
from iteblog;
tom
```

字符串函数

字符串长度函数：length
语法: length(string A)
返回值: int
说明：返回字符串A的长度

```
select length('abcedfg') from iteblog;
7
```

字符串反转函数：reverse
语法: reverse(string A)
返回值: string
说明：返回字符串A的反转结果

```
select reverse(abcedfg') from iteblog;
gfdecba
```

字符串连接函数：concat
语法: concat(string A, string B…)
返回值: string
说明：返回输入字符串连接后的结果，支持任意个输入字符串

```
select concat('abc','def','gh') from iteblog;
abcdefgh
```

带分隔符字符串连接函数：concat_ws
语法: concat_ws(string SEP, string A, string B…)
返回值: string
说明：返回输入字符串连接后的结果，SEP表示各个字符串间的分隔符

```
select concat_ws(',','abc','def','gh') from iteblog;
abc,def,gh
```

字符串截取函数：substr,substring
语法: substr(string A, int start),substring(string A, int start)
返回值: string
说明：返回字符串A从start位置到结尾的字符串

```
select substr('abcde',3) from iteblog;
cde
select substring('abcde',3) from iteblog;
cde
select substr('abcde',-1) from iteblog;   （和ORACLE相同）
e
```

字符串截取函数：substr,substring
语法: substr(string A, int start, int len),substring(string A, int start, int len)
返回值: string
说明：返回字符串A从start位置开始，长度为len的字符串

```
select substr('abcde',3,2) from iteblog;
cd
select substring('abcde',3,2) from iteblog;
cd
select substring('abcde',-2,2) from iteblog;
de
```

字符串转大写函数：upper,ucase
语法: upper(string A) ucase(string A)
返回值: string
说明：返回字符串A的大写格式

```
select upper('abSEd') from iteblog;
ABSED
select ucase('abSEd') from iteblog;
ABSED
```

字符串转小写函数：lower,lcase
语法: lower(string A) lcase(string A)
返回值: string
说明：返回字符串A的小写格式

```
select lower('abSEd') from iteblog;
absed
select lcase('abSEd') from iteblog;
absed
```

去空格函数：trim
语法: trim(string A)
返回值: string
说明：去除字符串两边的空格

```
select trim(' abc ') from iteblog;
abc
```

左边去空格函数：ltrim
语法: ltrim(string A)
返回值: string
说明：去除字符串左边的空格

```
select ltrim(' abc ') from iteblog;
abc
```

右边去空格函数：rtrim
语法: rtrim(string A)
返回值: string
说明：去除字符串右边的空格

```
select rtrim(' abc ') from iteblog;
abc
```

正则表达式替换函数：regexp_replace
语法: regexp_replace(string A, string B, string C)
返回值: string
说明：将字符串A中的符合java正则表达式B的部分替换为C。注意，在有些情况下要使用转义字符,类似oracle中的regexp_replace函数。

```
select regexp_replace('foobar', 'oo|ar', '') from iteblog;
fb
```

正则表达式解析函数：regexp_extract
语法: regexp_extract(string subject, string pattern, int index)
返回值: string
说明：将字符串subject按照pattern正则表达式的规则拆分，返回index指定的字符。

```
select regexp_extract('foothebar', 'foo(.*?)(bar)', 1) from iteblo
g;
the
select regexp_extract('foothebar', 'foo(.*?)(bar)', 2) from iteblo
g;
bar
select regexp_extract('foothebar', 'foo(.*?)(bar)', 0) from iteblo
g;
foothebar
```

注意，在有些情况下要使用转义字符，下面的等号要用双竖线转义，这是java正则表达式的规则。

```
select data_field,
  regexp_extract(data_field,'.*?bgStart\\=([^&]+)',1) as aaa,
  regexp_extract(data_field,'.*?contentLoaded_headStart\\=
([^&]+)',1) as bbb,
  regexp_extract(data_field,'.*?AppLoad2Req\\=([^&]+)',1) as ccc
  from pt_nginx_loginlog_st
  where pt = '2012-03-26' limit 2;
```

URL解析函数：parse_url
语法: parse_url(string urlString, string partToExtract [, string keyToExtract])
返回值: string
说明：返回URL中指定的部分。partToExtract的有效值为：HOST, PATH, QUERY, REF, PROTOCOL, AUTHORITY, FILE, and USERINFO.

```
select parse_url('https://www.iteblog.com/path1/p.php?k1=v1&k2=v2#Ref1', 'HOST') from iteblog;
facebook.com
select parse_url('https://www.iteblog.com/path1/p.php?k1=v1&k2=v2#Ref1', 'QUERY', 'k1') from iteblog;
v1
```

json解析函数：get_json_object
语法: get_json_object(string json_string, string path)
返回值: string
说明：解析json的字符串json_string,返回path指定的内容。如果输入的json字符串无效，那么返回NULL。

```
select  get_json_object('{"store":
   {"fruit":\[{"weight":8,"type":"apple"},{"weight":9,"type":"pear"}],
    "bicycle":{"price":19.95,"color":"red"}
   },
  "email":"amy@only_for_json_udf_test.net",
  "owner":"amy"
 }
 ','$.owner') from iteblog;
amy
```

空格字符串函数：space
语法: space(int n)
返回值: string
说明：返回长度为n的字符串

```
select space(10) from iteblog;
select length(space(10)) from iteblog;
10
```

重复字符串函数：repeat
语法: repeat(string str, int n)
返回值: string
说明：返回重复n次后的str字符串

```
select repeat('abc',5) from iteblog;
abcabcabcabcabc
```

首字符ascii函数：ascii
语法: ascii(string str)
返回值: int
说明：返回字符串str第一个字符的ascii码

```
select ascii('abcde') from iteblog;
97
```

左补足函数：lpad
语法: lpad(string str, int len, string pad)
返回值: string
说明：将str进行用pad进行左补足到len位

```
select lpad('abc',10,'td') from iteblog;
tdtdtdtabc
```

注意：与GP，ORACLE不同，pad 不能默认

右补足函数：rpad
语法: rpad(string str, int len, string pad)
返回值: string
说明：将str进行用pad进行右补足到len位

```
select rpad('abc',10,'td') from iteblog;
abctdtdtdt
```

分割字符串函数: split
语法: split(string str, string pat)
返回值: array
说明: 按照pat字符串分割str，会返回分割后的字符串数组

```
select split('abtcdtef','t') from iteblog;
["ab","cd","ef"]
```

集合查找函数: find_in_set
语法: find_in_set(string str, string strList)
返回值: int
说明: 返回str在strlist第一次出现的位置，strlist是用逗号分割的字符串。如果没有找该str字符，则返回0

```
select find_in_set('ab','ef,ab,de') from iteblog;
2
select find_in_set('at','ef,ab,de') from iteblog;
0
```

集合统计函数

个数统计函数: count
语法: count(*), count(expr), count(DISTINCT expr[, expr_.])
返回值: int
说明: count(*)统计检索出的行的个数，包括NULL值的行；count(expr)返回指定字段的非空值的个数；count(DISTINCT expr[, expr_.])返回指定字段的不同的非空值的个数

```
select count(*) from iteblog;
20
select count(distinct t) from iteblog;
10
```

总和统计函数: sum
语法: sum(col), sum(DISTINCT col)
返回值: double
说明: sum(col)统计结果集中col的相加的结果；sum(DISTINCT col)统计结果中col不同值相加的结果

```
select sum(t) from iteblog;
100
select sum(distinct t) from iteblog;
70
```

平均值统计函数: avg
语法: avg(col), avg(DISTINCT col)
返回值: double
说明: avg(col)统计结果集中col的平均值；avg(DISTINCT col)统计结果中col不同值相加的平均值

```
select avg(t) from iteblog;
50
select avg (distinct t) from iteblog;
30
```

最小值统计函数: min
语法: min(col)
返回值: double
说明: 统计结果集中col字段的最小值

```
select min(t) from iteblog;
20
```

最大值统计函数: max
语法: maxcol)
返回值: double
说明: 统计结果集中col字段的最大值

```
select max(t) from iteblog;
120
```

非空集合总体变量函数: var_pop
语法: var_pop(col)
返回值: double
说明: 统计结果集中col非空集合的总体变量（忽略null）

非空集合样本变量函数: var_samp
语法: var_samp (col)
返回值: double
说明: 统计结果集中col非空集合的样本变量（忽略null）

总体标准偏离函数: stddev_pop
语法: stddev_pop(col)
返回值: double
说明: 该函数计算总体标准偏离，并返回总体变量的平方根，其返回值与VAR_POP函数的平方根相同

样本标准偏离函数: stddev_samp
语法: stddev_samp (col)
返回值: double
说明: 该函数计算样本标准偏离

中位数函数: percentile
语法: percentile(BIGINT col, p)
返回值: double
说明: 求准确的第pth个百分位数，p必须介于0和1之间，但是col字段目前只支持整数，不支持浮点数类型

中位数函数: percentile
语法: percentile(BIGINT col, array(p1 [, p2]…))
返回值: array
说明: 功能和上述类似，之后后面可以输入多个百分位数，返回类型也为array，其中为对应的百分位数。

取0.2，0.4位置的数据`select percentile(score,&lt;0.2,0.4>) from iteblog;`

近似中位数函数: percentile_approx
语法: percentile_approx(DOUBLE col, p [, B])
返回值: double
说明: 求近似的第pth个百分位数，p必须介于0和1之间，返回类型为double，但是col字段支持浮点类型。参数B控制内存消耗的近似精度，B越大，结果的准确度越高。默认为10,000。当col字段中的distinct值的个数小于B时，结果为准确的百分位数

近似中位数函数: percentile_approx
语法: percentile_approx(DOUBLE col, array(p1 [, p2]…) [, B])
返回值: array
说明: 功能和上述类似，之后后面可以输入多个百分位数，返回类型也为array，其中为对应的百分位数。

直方图: histogram_numeric
语法: histogram_numeric(col, b)
返回值: array<struct {‘x’,‘y’}>
说明: 以b为基准计算col的直方图信息。

```
select histogram_numeric(100,5) from iteblog;
[{"x":100.0,"y":1.0}]
```

复合类型构建操作

Map类型构建: map
语法: map (key1, value1, key2, value2, …)
说明：根据输入的key和value对构建map类型

```
Create table iteblog as select map('100','tom','200','mary') as t f
rom iteblog;
describe iteblog;
t       map<string ,string>
hive> select t from iteblog;
{"100":"tom","200":"mary"}
```

Struct类型构建: struct
语法: struct(val1, val2, val3, …)
说明：根据输入的参数构建结构体struct类型

```
create table iteblog as select struct('tom','mary','tim') as t from
iteblog;
describe iteblog;
t       struct<col1:string ,col2:string,col3:string>
select t from iteblog;
{"col1":"tom","col2":"mary","col3":"tim"}
```

array类型构建: array
语法: array(val1, val2, ...)
说明：根据输入的参数构建数组array类型

```
create table iteblog as select array("tom","mary","tim") as t from
iteblog;
describe iteblog;
t       array
select t from iteblog;
["tom","mary","tim"]
```

复杂类型访问操作

array类型访问: A[n]
语法: A[n]
操作类型: A为array类型，n为int类型
说明：返回数组A中的第n个变量值。数组的起始下标为0。比如，A是个值为['foo', 'bar']的数组类型，那么A[0]将返回'foo',而A[1]将返回'bar'

```
create table iteblog as select array("tom","mary","tim") as t from
iteblog;
select t[0],t[1],t[2] from iteblog;
tom     mary    tim
```

map类型访问: M[key]
语法: M[key]
操作类型: M为map类型，key为map中的key值
说明：返回map类型M中，key值为指定值的value值。比如，M是值为{'f' -> 'foo', 'b' -> 'bar', 'all' -> 'foobar'}的map类型，那么M['all']将会返回'foobar'

```
Create table iteblog as select map('100','tom','200','mary') as t f
rom iteblog;
select t['200'],t['100'] from iteblog;
mary    tom
```

struct类型访问: S.x
语法: S.x
操作类型: S为struct类型
返回结构体S中的x字段。比如，对于结构体struct foobar {int foo, int bar}，foobar.foo返回结构体中的foo字段

```
create table iteblog as select struct('tom','mary','tim') as t from
iteblog;
describe iteblog;
t       struct<col1:string ,col2:string,col3:string>
select t.col1,t.col3 from iteblog;
tom     tim
```

复杂类型长度统计函数

Map类型长度函数: size(Map<k .V>)
语法: size(Map<k .V>)
返回值: int
说明: 返回map类型的长度

```
select size(map('100','tom','101','mary')) from iteblog;
2
```

array类型长度函数: size(Array `<T>`)
语法: size(Array `<T>`)
返回值: int
说明: 返回array类型的长度

```
select size(array('100','101','102','103')) from iteblog;
4
```

> **类型转换函数:** cast
> **语法:** cast(expr as `<type>`)
> **返回值:** Expected “=” to follow “type”
> **说明:** 返回转换后的数据类型

```
select cast(1 as bigint) from iteblog;
1
```

日志

日志信息结构分析

```
79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTT
P/1.1" 200 1671 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKi
t/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
```

1. 用户访问的ip地址
2. 用户标识
3. 用户名
4. 访问时间
5. 请求信息(请求方法，请求url，协议类型)
6. 请求相应状态
7. 请求相应流量大小(byte)
8. 关联页面
9. 客户端的浏览器类型

> 对数据进行分析之后，发现使用基本数据类型很难将数据给分隔开来，因此，我们使用特殊的数据类型进行数据存储

创建日志对应的表

```
CREATE TABLE apachelog (
  host STRING,
  identity STRING,
  username STRING,
  time STRING,
  request STRING,
  status STRING,
  size STRING,
  referer STRING,
  agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  "input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) (-|\\[[^\\]]*\\]) ([^
\"]*|\"[^\"]*\") (-|[0-9]*) (-|[0-9]*)(?: ([^ \"]*|\"[^\"]*\") ([^
\"]*|\"[^\"]*\"))?"
  ,"output.format.string"="%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s
%9$s"
)
STORED AS TEXTFILE;
```

加载数据

```
load data inpath '/log' overwrite into table apachelog
select * from apachelog
describe formatted apachelog
```

计算当日网站的pv uv

```
-- 计算当日网站的pv uv
-- pv 用户的每个请求就是一个pv；uv(一个ip就是uv的数量)
select count(*) pv
    ,count(distinct host) uv
from apachelog
```

统计出网站用户访问使用windows系统和使用mac系统的占比和数量

> 这个题目相对比较复杂，我们先将windows，mac，orther用户打上标签，然后，将标记过的数据放到一张临时表中，我们对这个临时表进行统计计算

- **匹配不同的用户**

```
select * from apachelog where agent rlike 'Windows NT'
select * from apachelog where agent rlike 'Mac OS'
```

- **创建临时表**

```
create table tmp_user_sys
stored as orc
as
select host
    ,sys_type
from (
select host
    ,case
        when agent rlike 'Windows NT' then 'windows'
        when agent rlike 'Mac OS' then 'mac'
        else 'other'
    end sys_type
from apachelog
) a
group by host
    ,sys_type
```

- 通过创建的临时表，我们看到同一用户有两台电脑，这里我们把既有mac又有windows的用户看做是两个不同的用户

```
select count(1) p_num
    ,sum(case when sys_type='mac' then 1 else 0 end) mac_num
    ,sum(case when sys_type='windows' then 1 else 0 end) windows_nu
m
    ,sum(case when sys_type='other' then 1 else 0 end) other_num
    ,sum(case when sys_type='mac' then 1 else 0 end)/count(1) mac_r
ate
    ,sum(case when sys_type='windows' then 1 else 0 end)/count(1) w
indows_rate
    ,sum(case when sys_type='other' then 1 else 0 end)/count(1) oth
er_rate
from tmp_user_sys
```

自定义hive function

> **自定义函数**，我们以时间格式转换为例

1. **编写时间转换格式代码**

```java
package top.xiesen.udf;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import org.apache.hadoop.hive.ql.exec.UDF;

/**
* 项目名称：udftest
* 类名称：LogDateConvert [14/Jun/2014:10:30:13 -0400] --》 2014-06-14 22:30:13
* 类描述：自定义udf需要继承UDF类，实现evaluate方法
* 返回值类型上，可以是java基础类型、writeable子类、string
* 创建人：Allen
* @version
*/
public class LogDateConvert extends UDF{
    public static final SimpleDateFormat FORMAT = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    // 输出数据的日期格式是带有local和时区的，local是英文，可以使用英文格式进行匹配转换
    public static final SimpleDateFormat SRCFORMAT = new SimpleDateFormat("[dd/MMM/yyyy:HH:mm:ss Z]",Locale.ENGLISH);


    /**
    * evaluate 将[14/Jun/2014:10:30:13 -0400] 格式转换成  2014-06-14 22:30:13
    * @param @param datestr [14/Jun/2014:10:30:13 -0400]格式的字符串
    * @return String 返回类型 ，可以是java基础类型、writeable子类、string
    * @Exception 异常对象
    */
    public String evaluate(String datestr){
        try {
            Date olDate = SRCFORMAT.parse(datestr);
            return FORMAT.format(olDate);
        } catch (ParseException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void main(String[] args) {
        LogDateConvert logDateConvert = new LogDateConvert();
        System.out.println(logDateConvert.evaluate("[14/Jun/2014:10:30:13 -0400]"));
```

```
    }
}
```

2. 将编写的代码，**打成jar包，上传到linux环境下**

```
add jar /root/udf.jar;
list jars;
```

3. **将jar包加载到hive上，创建function** `create function logdate_convert as 'top.xiesen.udf.LogDateConvert';`
4. **使用自定义函数**

```
-- 使用函数
create table hour_pvuv
stored as orc
as
select b.hour
    ,count(1) pv
    ,count(distinct b.host) uv
from (select hour(logdate_convert(time)) hour
    ,a.*
from apachelog a) b
group by b.hour
```

异常处理

1. 在创建hive function时出现 ClassNotFoundException异常信息
   解决方案：将jar包上传到hdfs上，再执行添加操作
2. 在查询结果信息时，出现查询结果不一致的现象
   解决方案：出现这种现象的原因是虚拟机上linux的时间是英国的时区，老师的系统将时区修改为东八区了，我们可以将linux系统上的时区修改一下，也可以将我们的代码修改了。出现这个问题，给我们的程序没有关系，也可以选择沉默。