Hive数据仓库中的ETL——基于Hive sql

一、使用正则表达式做数据清洗

1、基本语法

语法 解释

英文 A-Za-z

数字 0-9

() 标记一个子表达式的开始和结束的位置，括号内表示一个连续的表达式；如果要匹配小括号，需要用\转义，例如：()

[] 定义匹配的字符范围，例如：[a-zA-Z]表示匹配大小写英文字符

{} 定义匹配范围的长度，例如：{1}表示匹配一次，{1,}表示匹配大于等于1次，{2,3}表示匹配大于等于2小于等于3次

\* 匹配子表达式零次或多次

+ 匹配子表达式1次或多次

? 匹配子表达式0次或1次

\ 转义字符

^ 表示非的意思，例如：[^>]\* 表示非>的字符可以有0次或多次

\d 匹配一个数字字符，等价于[0-9]

\D 匹配一个非数字字符，等价于[^0-9]

\w 匹配字母、数字、下划线，等价于[A-Za-z0-9\_]

\W 匹配非字母、数字、下划线，等价于[^A-Za-z0-9\_]

. 匹配除换行符（\n、\r）之外的任何单个字符

2、最常用的正则清洗函数

2.1 REGEXP\_REPLACE()

string regexp\_replace(string <source>, string <pattern>, string <replace\_string>[, bigint <occurrence>])

命令说明

将source字符串中第occurrence次匹配pattern的子串替换成指定字符串replace\_string后返回结果字符串。

参数说明

source：必填。STRING类型，待替换的字符串。

pattern：必填。STRING类型常量或正则表达式。待匹配的模型。更多正则表达式编写规范，请参见正则表达式规范。pattern为空串时返回报错。

replace\_string：必填。STRING类型，将匹配pattern的字符串替换后的字符串。

occurrence：可选。BIGINT类型常量，必须大于等于0，表示将第occurrence次匹配的字符串替换为replace\_string，为0时表示替换所有匹配的子串。为其他类型或小于0时，返回报错。默认值为0。

实例说明

在做埋点访客量统计时，url后面有单号的需要做清洗，将单号去除，只留下url，例如

www.baidu.com/query/OO99238842934/RR99382333，清洗后的数据为：www.baidu.com/query

url = ‘www.baidu.com/query/OO99238842934/RR99382333’

regexp\_replace(url,'([\/]{1}[A-Z]{2,}[0-9]{5,})\*', '')

正则表达式中的()很重要说明括号内的是一个完整需要匹配的子表达式，如果没有小括号会认为需要匹配url中所有的/、大写字母和数字

SELECT

dt,

url,

(case

when site\_id in ('110077') then regexp\_replace(url,'([\/]{1}[A-Z]{2,}[0-9]{5,})\*', '')

else url

end) as url\_cleaned

FROM url\_info;

3、常用的正则清洗实例

注意：由于Hive是Java编写的,所以在编写Hive sql时，必须将所有\写成\\

3.1、手机号：

SELECT

CASE WHEN 手机号字段名 NOT REGEXP '^1(3[0-9]|4[01456879]|5[0-35-9]|6[2567]|7[0-8]|8[0-9]|9[0-35-9])\\d{8}$'

THEN NULL

ELSE 手机号字段名

END

FROM 表名

--上述是所有运营商号段写在一起的，包括虚拟运营商、物联网等

3.2、邮箱

由于邮箱的基本格式为“名称@域名”，需要使用“^”匹配邮箱的开始部分，用“$”匹配邮箱结束部分以保证邮箱前后不能有其他字符，

所以最终邮箱的正则表达式为：

^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$

SELECT CASE WHEN data.email NOT REGEXP '^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$' THEN NULL ELSE data.email END FROM ods\_user\_info\_inc ;

3.3、URL

域名：[a-zA-Z0-9][-a-zA-Z0-9]{0,62}(/.[a-zA-Z0-9][-a-zA-Z0-9]{0,62})+/.?

InternetURL：[a-zA-z]+://[^\s]\* 或 ^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]\*)?$

3.4、座机

(\d{3,4}[-])?\d{8}

3.5、中文名

^[\u4e00-\u9fa5]{2,4}$

3.6、地址

"\<([^省]+省)|([^自治区]+自治区)|([^自治州]+自治州)|([^市]+市)|([^盟]+盟)|([^地区]+地区)|([^区划]+区划)|([^县]+县)|([^旗]+旗)|([^区]+区)|([^镇]+镇)\>"

--只能匹配标准的省市县、市市县等类型的家庭地址

3.7、日期格式：^\d{4}-\d{1,2}-\d{1,2}

一年的12个月(01～09和1～12)：^(0?[1-9]|1[0-2])$

一个月的31天(01～09和1～31)：^((0?[1-9])|((1|2)[0-9])|30|31)$

3.8、金额：

有四种钱的表示形式我们可以接受:"10000.00" 和 "10,000.00", 和没有 "分" 的 "10000" 和 "10,000"

^([0-9]+|[0-9]{1,3}(,[0-9]{3})\*)(.[0-9]{1,2})?$

3.9、其他

身份证：(^\d{15}$)|(^\d{17}([0-9]|X|x)$)

IP地址:IP地址：((?:(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d)\\.){3}(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d))

\*匹配首尾空白字符的正则表达式：^s\*|s\*$

评注：可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等等)，非常有用的表达式

验证是否含有^%&',;=?$"等字符：“[^%&',;=?$x22]+”

匹配特定数字：

^[1-9]d\*$　 　 //匹配正整数

^-[1-9]d\*$ 　 //匹配负整数

^-?[1-9]d\*$　　 //匹配整数

^[1-9]d\*|0$　 //匹配非负整数（正整数 + 0）

^-[1-9]d\*|0$　　 //匹配非正整数（负整数 + 0）

^[1-9]d\*.d\*|0.d\*[1-9]d\*$　　 //匹配正浮点数

^-([1-9]d\*.d\*|0.d\*[1-9]d\*)$　 //匹配负浮点数

^-?([1-9]d\*.d\*|0.d\*[1-9]d\*|0?.0+|0)$　 //匹配浮点数

^[1-9]d\*.d\*|0.d\*[1-9]d\*|0?.0+|0$　　 //匹配非负浮点数（正浮点数 + 0）

^(-([1-9]d\*.d\*|0.d\*[1-9]d\*))|0?.0+|0$　　//匹配非正浮点数（负浮点数 + 0）

评注：处理大量数据时有用，具体应用时注意修正

二、常用的内置字符串函数

1、 case when 的利用，清洗诸如评分等的内容,用例如下。

case

when new.comment\_grade = '五星商户' then 50

when new.comment\_grade = '准五星商户' then 45

when new.comment\_grade = '四星商户' then 40

when new.comment\_grade = '准四星商户' then 35

when new.comment\_grade = '三星商户' then 30

when new.comment\_grade = '准三星商户' then 25

when new.comment\_grade = '二星商户' then 20

when new.comment\_grade = '准二星商户' then 15

when new.comment\_grade = '一星商户' then 10

when new.comment\_grade = '准一星商户' then 5

when new.comment\_grade = '该商户暂无星级' then 0

when new.comment\_grade is NULL then old.comment\_grade

else new.comment\_grade

END as `new.comment\_grade`,

2、 替换字符串中的一些内容(更详细可参考正则中提到的)

regexp\_replace(new.avg\_price, '-', '')

替换 avg\_price 中的中划线。

3、 字符串切分函数

split(a.tag\_flag, '>')[1],

具体例子：

select split('a,b', ',')[0] ===> 结果 a

4、 字符串拼接函数

select concat('1', '2', '3'); ===> 结果 123

select concat\_ws(':','name','hanmeimei')===>返回name:hanmeimei ;

select concat\_ws(':','avg',null,'34')===>返回NULL;

\*将ARRAY数组array('name', 'hanmeimei')中的元素通过:连接。命令示例如下

select concat\_ws(':',array('name', 'hanmeimei'))===>返回name:hanmeimei;

5、 去除字符串两端空格

trim(a.city)

ltrim,rtrim

6、 使用left join 或者 right join 、full join补全数据

例如根据两张表，其中一张表格table2含有省份和城市的信息，

其中一张表table1只有城市信息，需要补全table1 中的省份信息，可以像如下做法：

select

a.name,

b.province,

a.city

from table1 a left join table2 b on a.city = b.city;

7、过滤不符合条件的数据

可以使用等值判断来处理数据

清除一些不符合条件的数据。

INSERT OVERWRITE table ods.js\_beauty\_tmp

SELECT \*

from ods.js\_beauty\_tmp

WHERE map\_lat != ''

AND map\_lng != ''

AND map\_lat IS NOT NULL

AND map\_lng IS NOT NULL

AND map\_lat != 0

AND map\_lng != 0

AND map\_lat not like '-%'

AND map\_lng not like '-%'

and city != '其他城市'

and city != '点评实验室';

8、特殊函数

Array\_contains(a,b)

参数类型a数组，b String，判断数组a中是否包含b，返回boolean类型

Collect\_set()和Collect\_list(),参数是字符串，作用是将参数字段的所有值汇聚成一个数组，set可去重，list可排序，后边可以开窗

To\_data(),参数类型支持string,timestamp,作用是格式化日期---------'yyyy-MM-dd'

若输入非法日期，函数返回NULL(20220101)或者按公历逻辑向后计算(2022-12-31-----2023-01-01)