第6章Hive数据仓库实训

(源自:https://biglab.site)

(版本:Ver1.0-20231109)

实训1查询图书馆图书的被借阅次数及读者借阅图书的次数

实训要点

1. 掌握Hive数据表的创建及数据转载。
2. 掌握Hive中where、group by等查询操作方法

需求说明

借阅信息是挖掘读者信息行为模式的主要数据源，通过对它的统计、归类、挖掘分析有助于了解读者的偏好，对读者进行聚类细分，建立读者行为模型。某图书馆导出了一份读者借阅信息文件，数据字段说明如表6-7所示。请将数据保存至Hive数据仓库中，并使用HQL语句查询出被借阅最多的10本图书及借阅次数前10的读者信息

字段名称	说明	
序号	每一条借阅信息的序号	
索书号	图书的索引书号	
题名	图书名称	
读者姓名	读者姓名	
读者号	读者的卡号	
借出时间	该图书被读者借走的时间	

实现思路及步骤

1. 在Hive中创建一个borrow表，表结构与读者借阅信息文件的数据结构保持一致。
2. 将原始数据装载到borrow表中。
3. 将borrow表中的数据根据题名进行分组聚合，计算出每本图书的被借阅次数，再按降序排列，并取前10条数据，即可统计出被借阅最多的10本图书。
4. 将borrow表中的数据根据读者姓名进行分组聚合，计算出每个读者借阅图书的次数，再按降序排列，并取前10条数据，即可统计出借阅次数前10的读者信息.

作业要求

1. 在Hive中创建myname数据库，在myname库中创建borrow表，截图建表语句和运行结果；
2. 在Hive中将原始数据装载到borrow表中,截图装载脚本和运行过程；
3. 完成功能：将borrow表中的数据根据题名进行分组聚合，计算出每本图书的被借阅次数，再按降序排列，并取前10条数据，即可统计出被借阅最多的10本图书 ，截图分析脚本和运行过程；
4. 完成功能：将borrow表中的数据根据读者姓名进行分组聚合，计算出每个读者借阅图书的次数，再按降序排列，并取前10条数据，即可统计出借阅次数前10的读者信息，截图分析脚本和运行过程；

实现参考

数据准备

在master或slave上执行

```
cd /root/hadoop
wget https://biglab.site/b57562hadoop/file/hadoop_t_data.tar.gz
tar -zxvf ./hadoop_t_data.tar.gz
hdfs dfs -mkdir /user/myname/library
hdfs dfs -put ./hadoop_t_data/borrow.txt  /user/myname/library

```

数据分析

在master或slave的 hive命令行中执行：

```
-- 创建myname数据库，myname需要换成本人姓名拼音
create database myname;
-- 进入myname数据库
use myname;
```

```
-- 创建borrow表
create table borrow(
       id int,
       callNumber string,
       title string,
       name string,
       readNo string,
       borrowTime Timestamp)row format delimited fields terminated by
','stored as textfile;
-- 加载数据到borrow表中
load data inpath '/user/myname/library/borrow.txt' overwrite into table
borrow;
-- 将borrowInfo表中的数据根据书名进行分组聚合，计算出每本书的被借阅次数，再按降序排列，
并取前10条数据，即可统计出被借阅最多的前10本书
select title,num from (select title,count(title) as num,row_number()
over(order by count(title) DESC)as rn from borrow group by title)t where rn
<=10;
-- 将borrow表中的数据根据作者进行分组聚合，计算出每个读者的借阅书籍的次数，再按降序排
列，并取前10条数据，即可统计出借阅次数前10的读者信息。
select name,num from (select name,count(name) as num,row_number() over(order
by count(name) DESC)as rn from borrow group by name)t where rn <=10;
```

运行结果

```
hive> -- 创建borrow表
hive> create table borrow(
    >        id int,
    >        callNumber string,
    >        title string,
    >        name string,
    >        readNo string,
    >        borrowTime Timestamp)row format delimited fields terminated by
','stored as textfile;
OK
Time taken: 1.034 seconds
hive> -- 加载数据到borrow表中
hive> load data inpath '/user/myname/library/borrow.txt' overwrite into
table borrow;
Loading data to table default.borrow
OK
Time taken: 0.552 seconds
hive> -- 将borrowInfo表中的数据根据书名进行分组聚合，计算出每本书的被借阅次数，再按降
序排列，并取前10条数据，即可统计出被借阅最多的前10本书
hive> select title,num from (select title,count(title) as num,row_number()
over(order by count(title) DESC)as rn from borrow group by title)t where rn
<=10;
Query ID = root_20231110095300_1d925172-50dc-4802-8eb1-915288c20200
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

```
  set mapreduce.job.reduces=<number>
Starting Job = job_1699491849504_0011, Tracking URL =
http://master:8088/proxy/application_1699491849504_0011/
Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job  -kill
job_1699491849504_0011
Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 1
2023-11-10 09:53:15,609 Stage-1 map = 0%,  reduce = 0%
2023-11-10 09:53:23,832 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU
3.26 sec
2023-11-10 09:53:30,011 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU
5.42 sec
MapReduce Total cumulative CPU time: 5 seconds 420 msec
Ended Job = job_1699491849504_0011
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1699491849504_0012, Tracking URL =
http://master:8088/proxy/application_1699491849504_0012/
Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job  -kill
job_1699491849504_0012
Hadoop job information for Stage-2: number of mappers: 1; number of
reducers: 1
2023-11-10 09:53:44,354 Stage-2 map = 0%,  reduce = 0%
2023-11-10 09:53:53,750 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU
2.84 sec
2023-11-10 09:54:00,018 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU
6.1 sec
MapReduce Total cumulative CPU time: 6 seconds 100 msec
Ended Job = job_1699491849504_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.42 sec   HDFS Read:
3231637 HDFS Write: 1158617 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.1 sec   HDFS Read:
1169295 HDFS Write: 508 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 520 msec
OK
工程力学        109
概率论与数理统计        65
理论力学        62
线性代数        53
住区    53
自动控制原理    51
C语言程序设计   47
2       45
大学物理学      43
材料力学        38
Time taken: 61.774 seconds, Fetched: 10 row(s)
hive> -- 将borrow表中的数据根据作者进行分组聚合，计算出每个读者的借阅书籍的次数，再按
降序排列，并取前10条数据，即可统计出借阅次数前10的读者信息。
```

```
hive> select name,num from (select name,count(name) as num,row_number() over(order by count(name) DESC)as rn from borrow group by name)t where rn <=10;
Query ID = root_20231110095429_e46273a3-bb03-4319-9217-8e10b894673f
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1699491849504_0013, Tracking URL = http://master:8088/proxy/application_1699491849504_0013/
Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job  -kill job_1699491849504_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-11-10 09:54:36,434 Stage-1 map = 0%,  reduce = 0%
2023-11-10 09:54:43,566 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.15 sec
2023-11-10 09:54:48,736 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.44 sec
MapReduce Total cumulative CPU time: 4 seconds 440 msec
Ended Job = job_1699491849504_0013
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1699491849504_0014, Tracking URL = http://master:8088/proxy/application_1699491849504_0014/
Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job  -kill job_1699491849504_0014
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2023-11-10 09:55:01,840 Stage-2 map = 0%,  reduce = 0%
2023-11-10 09:55:10,116 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.68 sec
2023-11-10 09:55:16,436 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.35 sec
MapReduce Total cumulative CPU time: 4 seconds 350 msec
Ended Job = job_1699491849504_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.44 sec   HDFS Read: 3231740 HDFS Write: 276461 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.35 sec   HDFS Read: 287124 HDFS Write: 379 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 790 msec
OK
齐新勇  59
```

```
张杰    35
李伟    35
陈鹏    32
李涛    30
邓静静  29
刘涛    27
杜娟    27
刘强    27
周波    26
Time taken: 48.511 seconds, Fetched: 10 row(s)
```

环境搭建与实验视频

视频参考

实训2Hive实现表的连接及信息查询处理

实训要点

1. 掌握Hive数据表的创建及数据装载方法。
2. 熟悉Hive数据的插入语句。
3. 掌握Hive中where、group by、having、order by、join等查询操作方法。

需求说明

某小学的教学管理中均会使用表存放学生信息、课程信息、教师信息和学生成绩等信息。每个班级中都会存在student、course、teacher、score这4个表，分别对应student．txt、course．txt、teacher．txt、score．txt这4个数据文件。以其中一个班级为例，student．txt、course．txt、teacher．txt和score．txt文件的数据字段说明如表6-8所示

文件名称	字段名称	说明
student.txt	s_id	学生学号
	s_name	学生姓名
	s_birth	学生的出生日期
	s_sex	学生性别
course.txt	c_id	课程编号
	c_name	课程名称
	t_id	教师工号
teacher.txt	t_id	教师工号
	t_name	教师姓名
score.txt	s_id	学生学号
	c_id	课程编号
	s_score	学生该门课程的成绩

考虑到随着学校的发展，数据量会越来越多，可以使用Hive数据仓库存储数据。因此请设计出4个表对应的表结构并创建表，在Hive中分别实现以下需求:

1. 查询“01”课程比“02”课程成绩高的学生的信息及课程成绩。
2. 查询平均成绩大于等于60分的同学的学生学号、学生姓名和平均成绩。

实现思路及步骤

1. 根据4个数据文件的数据结构，分别创建 student、course、teacher、score这4个表。
2. 将本地数据加载至Hive表中。
3. 通过学生学号将student表和score表进行连接，再使用group by按学生学号和课程编号进行分组，利用where 语句查询“01”课程比“02”课程成绩高的学生的信息及课程成绩。
4. 将student表和score表进行连接，再按学生学号和学生姓名进行分组，接着对成绩进行avg聚合，最后利用having子句筛选平均成绩大于等于60分的学生信息。
5. 将 student和score表按学生学号进行左连接，按学生学号和学生姓名进行分组，在组内对课程编号进行统计聚合，并求分组后的每位同学的平均成绩。

作业要求

1. 在Hive中创建myname数据库，在myname库中创建student、course、teacher、score这4个表，截图建表语句和运行结果；
2. 在Hive中将原始数据装载到student、course、teacher、score表中,截图装载脚本和运行过程；
3. 完成功能：通过学生学号将student表和score表进行连接，再使用group by按学生学号和课程编号进行分组，利用where 语句查询“01”课程比“02”课程成绩高的学生的信息及课程成绩 ，截图分析脚本和运行过程；
4. 完成功能：将student表和score表进行连接，再按学生学号和学生姓名进行分组，接着对成绩进行avg聚合，最后利用having子句筛选平均成绩大于等于60分的学生信息，截图分析脚本和运行过程；

5. 完成功能：将 student和score表按学生学号进行左连接，按学生学号和学生姓名进行分组，在组内对课程编号进行统计聚合，并求分组后的每位同学的平均成绩，截图分析脚本和运行过程；

实现参考

数据准备

在master或slave上执行

```
cd /root/hadoop
wget https://biglab.site/b57562hadoop/file/hadoop_t_data.tar.gz
tar -zxvf ./hadoop_t_data.tar.gz
hdfs dfs -mkdir /user/myname/student
hdfs dfs -put ./hadoop_t_data/student.txt  /user/myname/student
hdfs dfs -put ./hadoop_t_data/course.txt  /user/myname/student
hdfs dfs -put ./hadoop_t_data/teacher.txt  /user/myname/student
hdfs dfs -put ./hadoop_t_data/score.txt  /user/myname/student
```

数据分析

在master或slave的 hive命令行中执行：

```
-- 创建student表
create table student(
    s_id string,
    s_name string,
    s_birth  string,
    s_sex string)row format delimited fields terminated by ','stored as textfile;
-- 创建course表
create table course (
    c_id string,
    c_name string,
    t_id string)row format delimited fields terminated by ','stored as textfile;
-- 创建teacher表
create table teacher (
    t_id string,
    t_name string)row format delimited fields terminated by ','stored as textfile;
-- 创建score表
create table score (
    s_id string,
    c_id string,
    s_score  int)row format delimited fields terminated by ','stored as textfile;

-- 将本地数据加载至Hive表中。
load data inpath '/user/myname/student/student.txt' overwrite into table student;
load data inpath '/user/myname/student/course.txt' overwrite into table course;
load data inpath '/user/myname/student/teacher.txt' overwrite into table teacher;
```

```
load data inpath '/user/myname/student/score.txt' overwrite into table score;
-- 查询“01”课程比“02”课程成绩高的学生的信息及课程分数。
select  student.*,a.s_score,b.s_score from student join score a on a.c_id = "01" join score b on b.c_id="02" where a.s_id =student.s_id and b.s_id = student.s_id and a.s_score > b.s_score;
-- 查询平均成绩大于等于60分的同学的学生编号和学生姓名和平均成绩。
select student.s_id,student.s_name,round(avg(score.s_score),1) from score join student on score.s_id = student.s_id group by student.s_id,student.s_name having avg(score.s_score)>60;
```

运行结果

```
hive> -- 创建student表
hive> create table student(
    >     s_id string,
    >     s_name string,
    >     s_birth  string,
    >     s_sex string)row format delimited fields terminated by ','stored as textfile;
OK
Time taken: 1.04 seconds
hive> -- 创建course表
hive> create table course (
    >     c_id string,
    >     c_name string,
    >     t_id string)row format delimited fields terminated by ','stored as textfile;
OK
Time taken: 0.078 seconds
hive> -- 创建teacher表
hive> create table teacher (
    >     t_id string,
    >     t_name string)row format delimited fields terminated by ','stored as textfile;
OK
Time taken: 0.074 seconds
hive> -- 创建score表
hive> create table score (
    >     s_id string,
    >     c_id string,
    >     s_score  int)row format delimited fields terminated by ','stored as textfile;
OK
Time taken: 0.052 seconds
hive>
    > -- 将本地数据加载至Hive表中。
    > load data inpath '/user/myname/student/student.txt' overwrite into table student;
Loading data to table default.student
OK
Time taken: 0.436 seconds
hive> load data inpath '/user/myname/student/course.txt' overwrite into table course;
Loading data to table default.course
```

```
OK
Time taken: 0.175 seconds
hive> load data inpath '/user/myname/student/teacher.txt' overwrite into table teacher;
Loading data to table default.teacher
OK
Time taken: 0.266 seconds
hive> load data inpath '/user/myname/student/score.txt' overwrite into table score;
Loading data to table default.score
OK
Time taken: 0.13 seconds
hive> -- 查询“01”课程比“02”课程成绩高的学生的信息及课程分数。
hive> select  student.*,a.s_score,b.s_score from student join score a on a.c_id = "01" join score b on b.c_id="02" where a.s_id =student.s_id and b.s_id = student.s_id and a.s_score > b.s_score;
No Stats for default@student, Columns: s_id, s_sex, s_name, s_birth
No Stats for default@score, Columns: s_id, c_id, s_score
No Stats for default@score, Columns: s_id, c_id, s_score
Query ID = root_20231110100550_8decb46b-c969-466b-9df8-5b83affb5a6f
Total jobs = 1
2023-11-10 10:06:02     Uploaded 1 File to: file:/tmp/root/52f84152-61b9-4f07-bf28-35ae00454f29/hive_2023-11-10_10-05-50_907_747319958880284024-1/-local-10004/HashTable-Stage-4/MapJoin-mapfile00--.hashtable (1601 bytes)
2023-11-10 10:06:02     Dump the side-table for tag: 1 with group count: 24 into file: file:/tmp/root/52f84152-61b9-4f07-bf28-35ae00454f29/hive_2023-11-10_10-05-50_907_747319958880284024-1/-local-10004/HashTable-Stage-4/MapJoin-mapfile01--.hashtable
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1699491849504_0015, Tracking URL = http://master:8088/proxy/application_1699491849504_0015/
Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job  -kill job_1699491849504_0015
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 0
2023-11-10 10:06:13,101 Stage-4 map = 0%,  reduce = 0%
2023-11-10 10:06:21,458 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 2.3 sec
MapReduce Total cumulative CPU time: 2 seconds 300 msec
Ended Job = job_1699491849504_0015
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1   Cumulative CPU: 2.3 sec   HDFS Read: 12684 HDFS Write: 929 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 300 msec
OK
2019213182      谭一卜  2003年12月      男      92      89
2019213180      田芯    2003年11月      女      88      87
2019213179      伍序友  2004年1月       女      96      83
2019213171      李永乐  2003年7月       女      96      81
2019213173      冯德龙  2003年3月       男      92      81
2019213178      何佳乐  2002年4月       女      97      81
2019213169      朱佳    2003年7月       女      89      80
```

```
2019213164      李华    2003年9月       女      95      79
2019213174      朱乐    2003年1月       女      93      79
2019213163      何伶    2003年9月       女      95      74
2019213161      赵凯    2003年5月       女      79      69
2019213172      萧景炎  2003年6月       男      70      68
Time taken: 31.663 seconds, Fetched: 12 row(s)
hive> -- 查询平均成绩大于等于60分的同学的学生编号和学生姓名和平均成绩。
hive> select student.s_id,student.s_name,round(avg(score.s_score),1) from
score join student on score.s_id = student.s_id group by
student.s_id,student.s_name having avg(score.s_score)>60;
Query ID = root_20231110100624_3652a842-c8ba-445b-9403-6680544354b5
Total jobs = 1
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1699491849504_0016, Tracking URL =
http://master:8088/proxy/application_1699491849504_0016/
Kill Command = /usr/local/hadoop-3.1.4/bin/mapred job  -kill
job_1699491849504_0016
Hadoop job information for Stage-2: number of mappers: 1; number of
reducers: 1
2023-11-10 10:06:43,067 Stage-2 map = 0%,  reduce = 0%
2023-11-10 10:06:51,532 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU
2.18 sec
2023-11-10 10:06:57,749 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU
4.68 sec
MapReduce Total cumulative CPU time: 4 seconds 680 msec
Ended Job = job_1699491849504_0016
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.68 sec   HDFS Read:
23441 HDFS Write: 1096 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 680 msec
OK
2019213160      张俊龙  88.8
2019213161      赵凯    80.8
2019213162      周萧    76.0
2019213163      何伶    86.8
2019213164      李华    88.8
2019213165      刘弦    84.0
2019213166      朱阳    83.6
2019213167      李念    79.8
2019213168      周森    83.2
2019213169      朱佳    81.6
2019213170      赵小伟  86.0
2019213171      李永乐  82.4
2019213172      萧景炎  75.6
2019213173      冯德龙  81.0
2019213174      朱乐    86.8
```

```
2019213176      陈小敏  78.6
2019213177      叶子俊  82.8
2019213178      何佳乐  82.8
2019213179      伍序友  85.0
2019213180      田芯    84.0
2019213181      潘扬    82.0
2019213182      谭一卜  79.4
2019213183      缪鹏飞  87.8
Time taken: 34.148 seconds, Fetched: 23 row(s)
hive>
```

版本历史

- Ver1.0-20231109
    - 初始版本