Reference：

[大数据软件安装和基础编程实践指南（2023年7月版）\_厦大数据库实验室博客](https://dblab.xmu.edu.cn/blog/4189/)

[Steam-Data-Analysis/spiders/spider\_pages.py at master · Xayanium/Steam-Data-Analysis](https://github.com/Xayanium/Steam-Data-Analysis/blob/master/spiders/spider_pages.py)

[精华报告：如何科学预估一款Steam游戏的销量？\_腾讯新闻](https://news.qq.com/rain/a/20231013A0B0H800)

[快速上手 - 使用手册 - Apache ECharts](https://echarts.apache.org/handbook/zh/get-started/)

[简介 · Bootstrap v4 中文文档 v4.6 | Bootstrap 中文网](https://v4.bootcss.com/docs/getting-started/introduction/)

hdfs dfs -mkdir -p /user/hive/warehouse/csvdata

hdfs dfs -put games.csv /user/hive/warehouse/csvdata/

插入csv：

CREATE EXTERNAL TABLE games (

id INT,

title STRING,

release\_date STRING,

review\_summary STRING,

discount FLOAT,

original\_price FLOAT,

final\_price FLOAT,

type\_list STRING,

developer STRING,

publisher STRING,

good\_rate FLOAT,

review\_count BIGINT,

overall\_evaluation STRING

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ",",

"quoteChar" = "\""

)

STORED AS TEXTFILE

LOCATION '/user/hive/warehouse/csvdata';

（其实hive从csv文件导入的数据均被默认设置为string，因为有杂质）

Mysql:

[虚拟机中的ubuntu的mysql怎么可以远程链接 - 知乎](https://zhuanlan.zhihu.com/p/404500556)

此时连接还会报错1045

原因是：MySQL 的连接方式不同，匹配的用户也不同！

当你使用命令：

mysql -u root -p

→ 默认连接的是 本地 socket 文件，MySQL 认为你是 'root'@'localhost'，所以能成功登录。

而当你用：

mysql -u root -p -h 192.168.133.132

→ 这是一个 TCP/IP 连接，MySQL 会尝试将你的主机名解析为 hadoop01/你的连接ip，然后查找是否有 'root'@'{你的连接ip}' 用户。

如果没有这个用户，它就会尝试 'root'@'%'。

但是！如果 MySQL 解析了你的主机名并发现有 'root'@'{你的连接ip}' 存在（即使你没显式创建），也可能导致权限错误或密码不匹配。

这里直接统一修改 'root'@'%的密码，让所有未注册的连接ip均使用一个密码连接

ALTER USER 'root'@'%' IDENTIFIED BY '165831';

FLUSH PRIVILEGES;

Sqoop教程：  
[Sqoop 安装配置（超详细）\_sqoop安装与配置-CSDN博客](https://blog.csdn.net/weixin_46389691/article/details/127663765)

注意sqoop连接mysql的ip不要用localhost，用hadoop01（自定义的ip映射）

传输时注意对应数据的类型以及编码格式（utf8mb4）正确一致，否则会报错

Mysql5.7的安全证书ssl与sqoop不兼容需修改参数为false

年份发布统计：

CREATE DATABASE IF NOT EXISTS game\_analysis;

USE game\_analysis;

Mysql：

CREATE TABLE IF NOT EXISTS publish\_year\_stats (

publish\_info VARCHAR(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci,

game\_count INT

);

Hivesql：

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/csvdata/publish\_year\_stats'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

SELECT

CASE

WHEN release\_date IS NOT NULL AND TRIM(release\_date) != '' THEN SUBSTR(release\_date, 1, 4)

ELSE NULL -- 如果是空或仅包含空白字符，则返回NULL

END AS publish\_info,

COUNT(\*) AS game\_count

FROM games

WHERE

release\_date IS NOT NULL

AND TRIM(release\_date) != '' -- 过滤掉空值和仅包含空白字符的记录

GROUP BY

CASE

WHEN release\_date IS NOT NULL AND TRIM(release\_date) != '' THEN SUBSTR(release\_date, 1, 4)

ELSE NULL

END;

转储实现csv：

hdfs dfs -cat /user/hive/warehouse/csvdata/year\_publish\_stats/\* > /tmp/year\_publish\_stats.csv

LOAD DATA LOCAL INFILE '/tmp/year\_publish\_stats.csv'

INTO TABLE year\_publish\_stats

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

(release\_year, game\_count)

SET release\_year = IF(@release\_year='', NULL, @release\_year),

game\_count = IF(@game\_count='', NULL, @game\_count);

Sqoop实现导出mysql：

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8" \

--username root \

--password 165831 \

--table publish\_year\_stats \

--export-dir /user/hive/warehouse/csvdata/publish\_year\_stats \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

年度热门统计：

Mysql：

CREATE TABLE IF NOT EXISTS yearly\_top\_games (

publish\_year INT,

title VARCHAR(255),

review\_count INT

);

Hivesql：

-- 第一步：创建临时表 valid\_games 处理原始数据并确保 review\_count 转换为数值类型

CREATE TEMPORARY TABLE IF NOT EXISTS valid\_games AS

SELECT

CASE

WHEN release\_date RLIKE '^[0-9]{4}-[0-1][0-9]-[0-3][0-9]$' THEN SUBSTR(release\_date, 1, 4) -- 提取日期前四位作为年份

ELSE NULL -- 对于非日期格式的数据设置为NULL

END AS publish\_year,

title,

CAST(CASE

WHEN review\_count RLIKE '^[0-9]+(\\.[0-9]+)?$' THEN review\_count -- 检查是否为数字格式

ELSE '0'

END AS DOUBLE) AS review\_count -- 将评论数转换为DOUBLE类型

FROM games

WHERE

(release\_date IS NOT NULL AND TRIM(release\_date) != '') -- 过滤掉空值和空白字符串

AND LENGTH(release\_date) >= 10 -- 简单过滤掉长度小于10的非日期格式数据

AND SUBSTR(release\_date, 1, 4) <= '2025'; -- 只保留2025年及之前的记录;

-- 第二步：创建临时表 ranked\_games 进行排名

CREATE TEMPORARY TABLE IF NOT EXISTS ranked\_games AS

SELECT

CAST(publish\_year AS INT) AS publish\_year,

title,

review\_count,

ROW\_NUMBER() OVER (PARTITION BY publish\_year ORDER BY review\_count DESC) AS rank

FROM valid\_games

WHERE publish\_year IS NOT NULL; -- 只考虑有有效年份的数据;

-- 第三步：导出排名前五的游戏信息到 /user/hive/warehouse/yearly\_top\_games

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/yearly\_top\_games'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

SELECT publish\_year, title, review\_count

FROM ranked\_games

WHERE rank <= 5 -- 只取前五名

ORDER BY publish\_year DESC, review\_count DESC;

Hdfs dfs -cat /user/hive/warehouse/yearly\_top\_games/\*

-- 清理临时表

DROP TABLE IF EXISTS valid\_games;

DROP TABLE IF EXISTS ranked\_games;

Sqoop：  
sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8" \

--username root \

--password 165831 \

--table yearly\_top\_games \

--export-dir /user/hive/warehouse/yearly\_top\_games \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

近一年内发布游戏的表现：

Mysql：

CREATE TABLE IF NOT EXISTS recent\_games\_analysis (

title VARCHAR(255) NOT NULL, -- 游戏标题

release\_date DATE, -- 发布日期

review\_count\_num DOUBLE, -- 转换后的评论数量

good\_rate VARCHAR(10), -- 好评率

original\_price VARCHAR(50), -- 原价

final\_price VARCHAR(50), -- 现价

developer VARCHAR(255), -- 开发者

publisher VARCHAR(255), -- 发行商

overall\_evaluation VARCHAR(255), -- 整体评价

rank\_by\_reviews INT -- 根据评论数排名

);

Hivesql：

-- Step 1: 创建临时表recent\_games，仅包含最近一年内发布的游戏。

-- Step 1: 创建临时表 recent\_games，仅包含最近一年内发布的游戏

CREATE TEMPORARY TABLE IF NOT EXISTS recent\_games AS

SELECT

title, -- 游戏标题

CASE

WHEN release\_date RLIKE '^[0-9]{4}-[0-1][0-9]-[0-3][0-9]$' THEN TO\_DATE(release\_date)

ELSE NULL

END AS release\_date,

CAST(CASE

WHEN review\_count RLIKE '^[0-9]+(\\.[0-9]+)?$' THEN review\_count

ELSE '0'

END AS DOUBLE) AS review\_count\_num,

good\_rate,

original\_price,

final\_price,

developer,

publisher,

overall\_evaluation

FROM games

WHERE

release\_date IS NOT NULL

AND release\_date RLIKE '^[0-9]{4}-[0-1][0-9]-[0-3][0-9]$' -- 确保日期格式正确

AND TO\_DATE(release\_date) >= date\_add(current\_date(), -365); -- 取近一年的数据

-- Step 2: 添加排名信息

CREATE TEMPORARY TABLE IF NOT EXISTS recent\_games\_performance AS

SELECT

title,

release\_date,

review\_count\_num,

good\_rate,

original\_price,

final\_price,

developer,

publisher,

overall\_evaluation,

ROW\_NUMBER() OVER (ORDER BY review\_count\_num DESC) AS rank\_by\_reviews

FROM recent\_games;

-- Step 3: 导出结果到 HDFS

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/recent\_games\_analysis'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM recent\_games\_performance ORDER BY rank\_by\_reviews;

DROP TABLE IF EXISTS recent\_games;

DROP TABLE IF EXISTS recent\_games\_performance;

Sqoop：

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table recent\_games\_analysis \

--export-dir /user/hive/warehouse/recent\_games\_analysis \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

高好评率游戏统计：

评论数>50000

Mysql：

CREATE TABLE IF NOT EXISTS high\_rating\_popular\_games (

title VARCHAR(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci,

release\_date DATE,

good\_rate\_num DOUBLE,

review\_count\_num DOUBLE,

original\_price VARCHAR(50),

final\_price VARCHAR(50),

developer VARCHAR(255),

publisher VARCHAR(255),

overall\_evaluation VARCHAR(255)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

Hivesql：

-- Step 1: 创建临时表 high\_rating\_popular\_games，筛选高好评 + 高评论数游戏（历史所有）

CREATE TEMPORARY TABLE IF NOT EXISTS high\_rating\_popular\_games AS

SELECT

title, -- 游戏标题

TO\_DATE(release\_date) AS release\_date, -- 发布日期

CAST(REGEXP\_REPLACE(good\_rate, '%', '') AS DOUBLE) AS good\_rate\_num, -- 好评率转成数值

CAST(

CASE

WHEN review\_count RLIKE '^([0-9]+\\.?[0-9]\*)$' THEN review\_count -- 支持整数和小数

ELSE '0'

END AS DOUBLE

) AS review\_count\_num, -- 评论数转成数值

original\_price,

final\_price,

developer,

publisher,

overall\_evaluation

FROM (

SELECT

title,

release\_date,

good\_rate,

review\_count,

original\_price,

final\_price,

developer,

publisher,

overall\_evaluation

FROM games

WHERE

release\_date RLIKE '^[0-9]{4}-[0-1][0-9]-[0-3][0-9]$' -- 确保发布日期格式为 YYYY-MM-DD

AND COALESCE(review\_count, '0') RLIKE '^[0-9]+(\\.[0-9]+)?$' -- 只保留数字或浮点数

AND CAST(REGEXP\_REPLACE(good\_rate, '%', '') AS DOUBLE) > 90 -- 好评率 > 90%

AND CAST(review\_count AS DOUBLE) > 50000 -- 评论数 > 50000

) t;

-- Step 2: 导出到 HDFS

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/high\_rating\_popular\_games'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM high\_rating\_popular\_games;

Sqoop：

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table high\_rating\_popular\_games \

--export-dir /user/hive/warehouse/high\_rating\_popular\_games \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

评价与游戏类型关系分析：

评论数>10000

Mysql：

CREATE TABLE IF NOT EXISTS game\_type\_analysis (

type VARCHAR(100) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci,

game\_count INT,

avg\_good\_rate DOUBLE,

total\_review\_count double

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

Hivesql：

-- Step 1: 创建临时表 game\_type\_analysis

CREATE TEMPORARY TABLE IF NOT EXISTS game\_type\_analysis AS

SELECT

type,

COUNT(DISTINCT title) AS game\_count,

AVG(good\_rate\_num) AS avg\_good\_rate,

SUM(review\_count\_num) AS total\_review\_count

FROM (

SELECT

title,

type,

CAST(REGEXP\_REPLACE(good\_rate, '%', '') AS DOUBLE) AS good\_rate\_num,

CAST(review\_count AS DOUBLE) AS review\_count\_num

FROM games

LATERAL VIEW explode(split(type\_list, ',')) typeTable as type

WHERE

type\_list IS NOT NULL AND

type\_list != '' AND

good\_rate RLIKE '^[0-9]+(\\.[0-9]\*)?$' AND

review\_count RLIKE '^[0-9]+(\\.[0-9]\*)?$'

) exploded\_games

GROUP BY type

HAVING total\_review\_count > 10000

ORDER BY avg\_good\_rate DESC;

-- Step 2: 导出结果到 HDFS

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/game\_type\_analysis'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM game\_type\_analysis;

Sqoop：

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table game\_type\_analysis \

--export-dir /user/hive/warehouse/game\_type\_analysis \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

游戏价格区间分布统计：

Mysql：

CREATE TABLE IF NOT EXISTS game\_price\_distribution (

price\_range VARCHAR(50),

game\_count INT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

Hivesql：

-- Step 1: 创建临时表 game\_price\_distribution

CREATE TEMPORARY TABLE IF NOT EXISTS game\_price\_distribution AS

SELECT

CASE

WHEN original\_price\_num = 0 THEN 'Free'

WHEN original\_price\_num > 0 AND original\_price\_num <= 10 THEN '0 - 10'

WHEN original\_price\_num > 10 AND original\_price\_num <= 20 THEN '10 - 20'

WHEN original\_price\_num > 20 AND original\_price\_num <= 30 THEN '20 - 30'

WHEN original\_price\_num > 30 AND original\_price\_num <= 40 THEN '30 - 40'

WHEN original\_price\_num > 40 AND original\_price\_num <= 50 THEN '40 - 50'

WHEN original\_price\_num > 50 AND original\_price\_num <= 100 THEN '50 - 100'

WHEN original\_price\_num > 100 AND original\_price\_num <= 200 THEN '100 - 200'

WHEN original\_price\_num > 200 AND original\_price\_num <= 400 THEN '200 - 400'

WHEN original\_price\_num > 400 AND original\_price\_num <= 800 THEN '400 - 800'

WHEN original\_price\_num > 800 THEN '800+'

ELSE 'Unknown' -- 处理无法识别的价格

END AS price\_range,

COUNT(\*) AS game\_count -- 统计每个区间的游戏数量

FROM (

SELECT

CAST(original\_price AS DOUBLE) AS original\_price\_num

FROM games

WHERE

original\_price RLIKE '^[0-9]+(\\.[0-9]\*)?$' -- 确保 original\_price 是有效的数字或小数

) t

GROUP BY

CASE

WHEN original\_price\_num = 0 THEN 'Free'

WHEN original\_price\_num > 0 AND original\_price\_num <= 10 THEN '0 - 10'

WHEN original\_price\_num > 10 AND original\_price\_num <= 20 THEN '10 - 20'

WHEN original\_price\_num > 20 AND original\_price\_num <= 30 THEN '20 - 30'

WHEN original\_price\_num > 30 AND original\_price\_num <= 40 THEN '30 - 40'

WHEN original\_price\_num > 40 AND original\_price\_num <= 50 THEN '40 - 50'

WHEN original\_price\_num > 50 AND original\_price\_num <= 100 THEN '50 - 100'

WHEN original\_price\_num > 100 AND original\_price\_num <= 200 THEN '100 - 200'

WHEN original\_price\_num > 200 AND original\_price\_num <= 400 THEN '200 - 400'

WHEN original\_price\_num > 400 AND original\_price\_num <= 800 THEN '400 - 800'

WHEN original\_price\_num > 800 THEN '800+'

ELSE 'Unknown'

END;

-- Step 2: 导出结果到 HDFS

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/game\_price\_distribution'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM game\_price\_distribution;

Sqoop：

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table game\_price\_distribution \

--export-dir /user/hive/warehouse/game\_price\_distribution \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

游戏价格与好评率分析：

Mysql：

CREATE TABLE IF NOT EXISTS game\_price\_goodrate\_analysis (

price\_range VARCHAR(50),

game\_count INT,

avg\_good\_rate DOUBLE,

total\_review\_count double

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

Hivesql：

-- Step 1: 创建临时表 game\_price\_goodrate\_analysis

CREATE TEMPORARY TABLE IF NOT EXISTS game\_price\_goodrate\_analysis AS

SELECT

CASE

WHEN price\_num = 0 THEN 'Free'

WHEN price\_num > 0 AND price\_num <= 10 THEN '0 - 10'

WHEN price\_num > 10 AND price\_num <= 20 THEN '10 - 20'

WHEN price\_num > 20 AND price\_num <= 30 THEN '20 - 30'

WHEN price\_num > 30 AND price\_num <= 40 THEN '30 - 40'

WHEN price\_num > 40 AND price\_num <= 50 THEN '40 - 50'

WHEN price\_num > 50 AND price\_num <= 100 THEN '50 - 100'

WHEN price\_num > 100 AND price\_num <= 200 THEN '100 - 200'

WHEN price\_num > 200 AND price\_num <= 400 THEN '200 - 400'

WHEN price\_num > 400 AND price\_num <= 800 THEN '400 - 800'

WHEN price\_num > 800 THEN '800+'

ELSE 'Unknown'

END AS price\_range,

COUNT(\*) AS game\_count,

AVG(good\_rate\_num) AS avg\_good\_rate,

SUM(review\_count\_num) AS total\_review\_count

FROM (

SELECT

CAST(original\_price AS DOUBLE) AS price\_num,

CAST(REGEXP\_REPLACE(good\_rate, '%', '') AS DOUBLE) AS good\_rate\_num,

CAST(review\_count AS DOUBLE) AS review\_count\_num

FROM games

WHERE

original\_price RLIKE '^[0-9]+(\\.[0-9]\*)?$' AND

good\_rate RLIKE '^[0-9]+(\\.[0-9]\*)?$' AND

review\_count RLIKE '^[0-9]+(\\.[0-9]\*)?$'

) t

GROUP BY

CASE

WHEN price\_num = 0 THEN 'Free'

WHEN price\_num > 0 AND price\_num <= 10 THEN '0 - 10'

WHEN price\_num > 10 AND price\_num <= 20 THEN '10 - 20'

WHEN price\_num > 20 AND price\_num <= 30 THEN '20 - 30'

WHEN price\_num > 30 AND price\_num <= 40 THEN '30 - 40'

WHEN price\_num > 40 AND price\_num <= 50 THEN '40 - 50'

WHEN price\_num > 50 AND price\_num <= 100 THEN '50 - 100'

WHEN price\_num > 100 AND price\_num <= 200 THEN '100 - 200'

WHEN price\_num > 200 AND price\_num <= 400 THEN '200 - 400'

WHEN price\_num > 400 AND price\_num <= 800 THEN '400 - 800'

WHEN price\_num > 800 THEN '800+'

ELSE 'Unknown'

END;

-- Step 2: 导出结果到 HDFS

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/game\_price\_goodrate\_analysis'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM game\_price\_goodrate\_analysis;

Sqoop;

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table game\_price\_goodrate\_analysis \

--export-dir /user/hive/warehouse/game\_price\_goodrate\_analysis \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

游戏销售额分析：

Mysql

CREATE TABLE IF NOT EXISTS top\_1000\_games\_by\_sales (

title VARCHAR(255), -- 假设游戏标题最长为255字符

original\_price DOUBLE,

review\_count double,

sales DOUBLE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

Hivesql

-- Step 1: 创建临时表 game\_sales 计算销售额并带上 title

CREATE TEMPORARY TABLE IF NOT EXISTS game\_sales AS

SELECT

title, -- 加入游戏标题

original\_price,

review\_count,

(CAST(review\_count AS DOUBLE) \* 30 \* CAST(original\_price AS DOUBLE)) AS sales -- 销售额计算

FROM games;

-- Step 2: 对游戏销售额进行排序，选出前1000个，并带上 title

CREATE TEMPORARY TABLE IF NOT EXISTS top\_1000\_games\_by\_sales AS

SELECT

title, -- 包含游戏标题

original\_price,

review\_count,

sales

FROM game\_sales

ORDER BY sales DESC

LIMIT 1000;

-- Step 3: 导出结果到 HDFS

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/top\_1000\_games\_by\_sales'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM top\_1000\_games\_by\_sales;

Sqoop:

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table top\_1000\_games\_by\_sales \

--export-dir /user/hive/warehouse/top\_1000\_games\_by\_sales \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

开发商好评率统计:

（好评数）

Mysql:

CREATE TABLE IF NOT EXISTS top\_developers\_by\_goodcount (

developer VARCHAR(255), -- 假设开发商名称最长为255字符

game\_count BIGINT, -- 游戏数量

avg\_good\_rate DOUBLE, -- 平均好评率

total\_good\_reviews DOUBLE -- 好评数总和

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

Hivesql:

-- Step 1: 创建临时表 developer\_goodcount

CREATE TEMPORARY TABLE IF NOT EXISTS developer\_goodcount AS

SELECT

developer,

COUNT(\*) AS game\_count,

SUM(

CAST(REGEXP\_REPLACE(good\_rate, '%', '') AS DOUBLE) \*

CAST(review\_count AS DOUBLE)

) AS total\_good\_reviews,

AVG(

CAST(REGEXP\_REPLACE(good\_rate, '%', '') AS DOUBLE)

) AS avg\_good\_rate

FROM games

WHERE

good\_rate IS NOT NULL AND good\_rate <> '' AND

review\_count IS NOT NULL AND review\_count <> ''

GROUP BY developer;

-- Step 2: 按照好评数总和降序排列开发商

CREATE TEMPORARY TABLE IF NOT EXISTS top\_developers\_by\_goodcount AS

SELECT

developer,

game\_count,

avg\_good\_rate,

total\_good\_reviews

FROM developer\_goodcount

ORDER BY total\_good\_reviews DESC;

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/top\_developers\_by\_goodcount'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM top\_developers\_by\_goodcount;

Sqoop：

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table top\_developers\_by\_goodcount \

--export-dir /user/hive/warehouse/top\_developers\_by\_goodcount \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

开发商销售额分析：

Mysql：

CREATE TABLE IF NOT EXISTS top\_developers\_by\_sales (

developer VARCHAR(255),

game\_count BIGINT,

avg\_good\_rate DOUBLE,

total\_sales DOUBLE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

Hivesql：

-- Step 1: 创建临时表 developer\_sales 统计每个开发商的总销售额

CREATE TEMPORARY TABLE IF NOT EXISTS developer\_sales AS

SELECT

developer,

COUNT(\*) AS game\_count,

SUM(

CAST(review\_count AS DOUBLE) \* 30 \* CAST(original\_price AS DOUBLE)

) AS total\_sales,

AVG(

CAST(REGEXP\_REPLACE(good\_rate, '%', '') AS DOUBLE)

) AS avg\_good\_rate

FROM games

WHERE

review\_count IS NOT NULL AND review\_count <> '' AND

original\_price IS NOT NULL AND original\_price <> ''

GROUP BY developer;

-- Step 2: 按照 total\_sales 排序并选出前100个

CREATE TEMPORARY TABLE IF NOT EXISTS top\_developers\_by\_sales AS

SELECT

developer,

game\_count,

avg\_good\_rate,

total\_sales

FROM developer\_sales

ORDER BY total\_sales DESC

LIMIT 100;

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/top\_developers\_by\_sales'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

SELECT \* FROM top\_developers\_by\_sales;

Sqoop：

sqoop export \

--connect "jdbc:mysql://hadoop01:3306/game\_analysis?useSSL=false&characterEncoding=UTF-8&useUnicode=true" \

--username root \

--password 165831 \

--table top\_developers\_by\_sales \

--export-dir /user/hive/warehouse/top\_developers\_by\_sales \

--input-fields-terminated-by '\t' \

--input-null-string '\\N' \

--input-null-non-string '\\N' \

-m 1

AI

我现在在做一个大数据的课设，已经完成了数据结果的查询分析并存储到了mysql当中，也完成了基础的数据传输接口，就差前端的展示了，接下来请你在静态资源中完成前端结合echarts对数据用图表的形式的展示，但是现在我们先做一个欢迎页面，要求炫酷且美观，同时给出展示如下信息的方案（跳转页面还是在一个页面修改容器展示）：

§4.2 发布年份统计 22

§4.2.1 年份发布统计 22

§4.2.2 年度热门游戏统计 22

§4.2.3 近一年内发布游戏的表现 22

§4.3 评价分析 22

§4.3.1 高好评率游戏统计 22

§4.3.2 评价与游戏类型关系分析 22

§4.4 价格分析 22

§4.4.1 游戏价格区间分布统计 22

§4.4.2 游戏价格与好评率分析 22

§4.4.3 游戏销售额分析 22

§4.5 开发商分析 22

§4.5.1 开发商好评率统计 22

§4.5.2 开发商销售额分析 22

接口已给出