《数据库原理及应用A》

（2024-2025学年第1学期）

实

验

报

告

学号：2023327100056

姓名：李凯涛

班级：23计算机科学与技术4班

实验三 DBMS表查询操作

目标1：满分70分 目标2：满分30分

### 一、实验目的：

* 1. 熟悉DBMS实验环境；
  2. 掌握SQL语言对数据库进行操作的方法，判断SQL语句运行的结果的正确性；
  3. 掌握图形化交互工具、命令行等工具的基本操作，学会数据库及表的创建与管理；
  4. 理解和掌握SQL语言:能够使用select语句进行符合要求的查询操作。

### 二、实验要求：

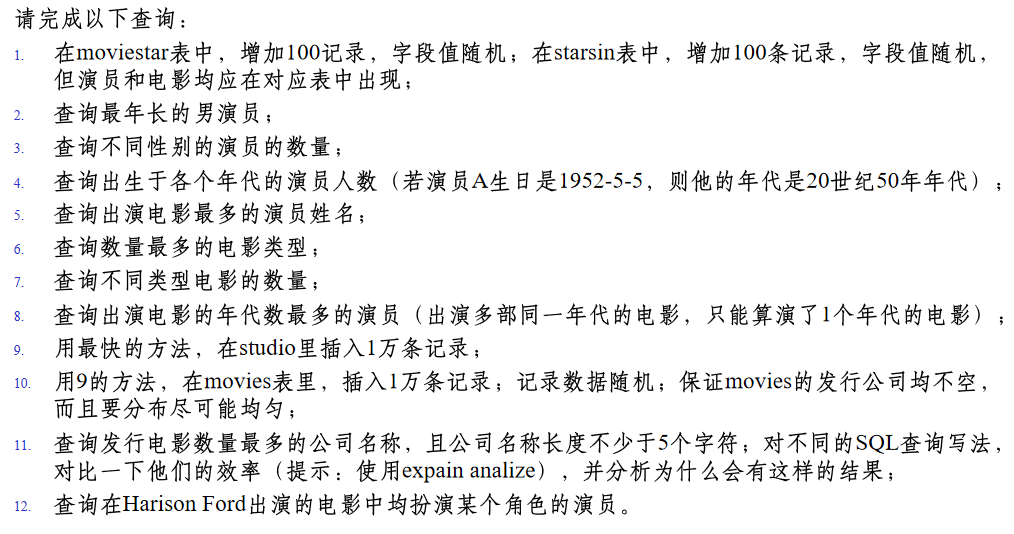
* 1. 熟悉系统数据库的结构握SQL语句常见语法错误的调试方法。
  2. 熟悉样例数据库的内容。

### 三、实验重点和难点：

实验重点：多表查询，嵌套查询，聚集函数的使用；

实验难点：SQL实现除法。

### 四、实验内容一：



完成上图中的1-8，11、12题，每个题的结果均应使用select查询结果来显示，其中第12题，应自行增加一些记录来显示SQL语句的正确性。

（1）在moviestar表中,增加100记录,字段值随机;在starsin表中,增加100条记录,字段值随机,

但演员和电影均应在对应表中出现;

知晓思路是要产生100条相关记录，可以使用MySQL直接插入相关语句，可以让AI进行生成

或用python循环语句生成：

import random

# 定义一些常量

titles = [f"Movie {chr(65 + i)}" for i in range(100)]

years = list(range(2010, 2024))

lengths = list(range(90, 150))

movieTypes = ["Drama", "Comedy", "Action", "Thriller", "Romance"]

studios = [f"Studio {i}" for i in range(1, 6)]

producers = list(range(1, 6))

stars = ["Harrison Ford", "Carrie Fisher", "Mark Hamill", "Debra Winger", "Jack Nicholson", "Kevin Spacey"]

# 生成 movies 表的数据

movies\_data = []

for i in range(100):

title = titles[i]

year = random.choice(years)

length = random.choice(lengths)

movieType = random.choice(movieTypes)

studio = random.choice(studios)

producer = random.choice(producers)

movies\_data.append((title, year, length, movieType, studio, producer))

# 生成 starsin 表的数据

starsin\_data = []

for i in range(100):

title = titles[i]

year = movies\_data[i][1] # 对应 movies 表中的 year

star = random.choice(stars)

starsin\_data.append((title, year, star))

# 生成 SQL 语句

movies\_sql = "INSERT INTO movies (title, year, length, movieType, studioName, producerC) VALUES\n"

starsin\_sql = "INSERT INTO starsin (movieTitle, movieYear, starName) VALUES\n"

for i, data in enumerate(movies\_data):

movies\_sql += f"('{data[0]}', {data[1]}, {data[2]}, '{data[3]}', '{data[4]}', {data[5]})"

if i < 99:

movies\_sql += ",\n"

else:

movies\_sql += ";\n"

for i, data in enumerate(starsin\_data):

starsin\_sql += f"('{data[0]}', {data[1]}, '{data[2]}')"

if i < 99:

starsin\_sql += ",\n"

else:

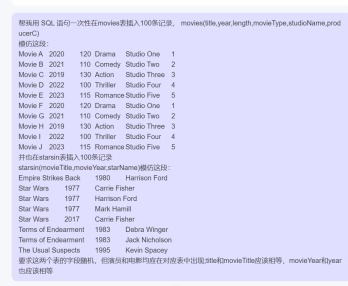
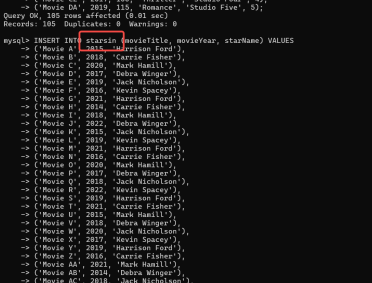
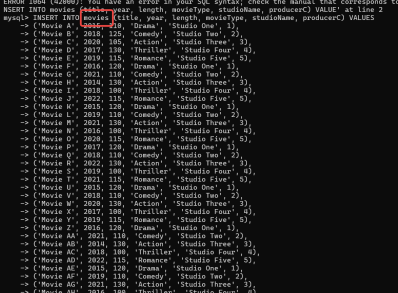
starsin\_sql += ";\n"

# 输出 SQL 语句

print(movies\_sql)

print(starsin\_sql)

截图：



1. 查询最年长的男演员;

语句：

SELECT `name`,address,gender,birthdate

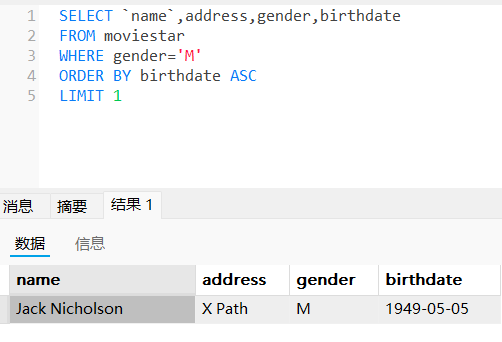
FROM moviestar

WHERE gender='M'

ORDER BY birthdate ASC

LIMIT 1

截图：



1. 查询不同性别的演员的数量;

语句：

SELECT gender,COUNT(\*)AS peoplecount

FROM moviestar

GROUP BY gender;

截图：



（4）查询出生于各个年代的演员人数(若演员A生日是1952-5-5,则他的年代是20世纪50年年代);

语句：

SELECT

CASE

WHEN YEAR(birthdate)BETWEEN 1940 AND 1949 THEN'20世纪40年年代'

WHEN YEAR(birthdate)BETWEEN 1950 AND 1959 THEN'20世纪50年年代'

WHEN YEAR(birthdate)BETWEEN 1970 AND 1979 THEN'20世纪70年年代'

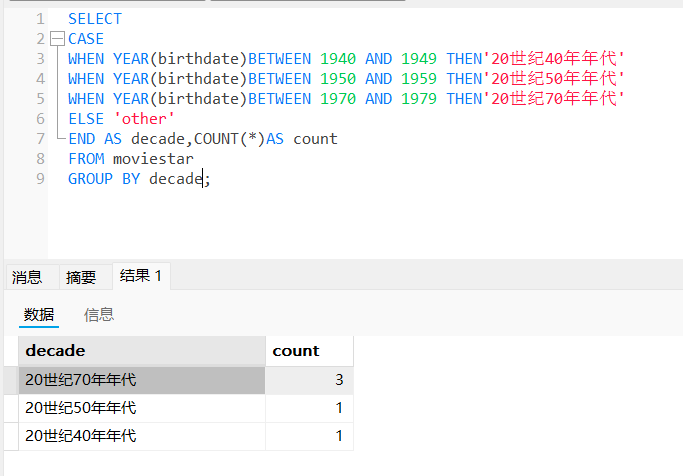
ELSE 'other'

END AS decade,COUNT(\*)AS count

FROM moviestar

GROUP BY decade;

截图：



（5）查询出演电影最多的演员姓名;

语句：

SELECT starsin.starName,COUNT(starsin.movieTitle)AS movie\_count

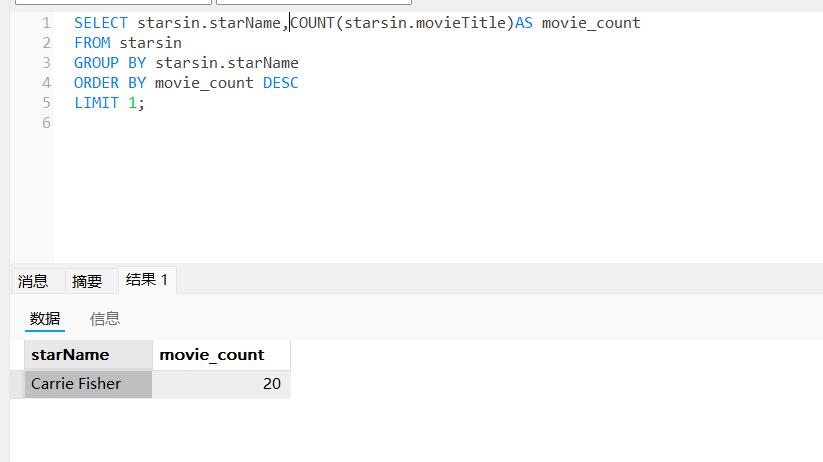
FROM starsin

GROUP BY starsin.starName

ORDER BY movie\_count DESC

LIMIT 1;

截图：



（6）查询数量最多的电影类型;

语句：

SELECT movies.movieType,COUNT(movies.movieType)AS movie\_count

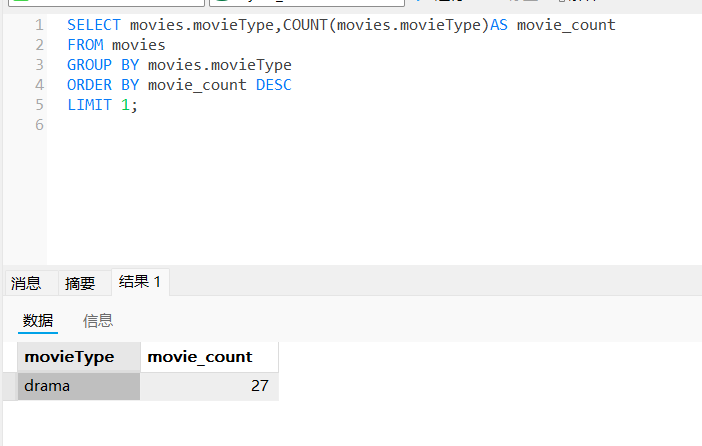
FROM movies

GROUP BY movies.movieType

ORDER BY movie\_count DESC

LIMIT 1;

截图：



（7）查询不同类型电影的数量;

语句：

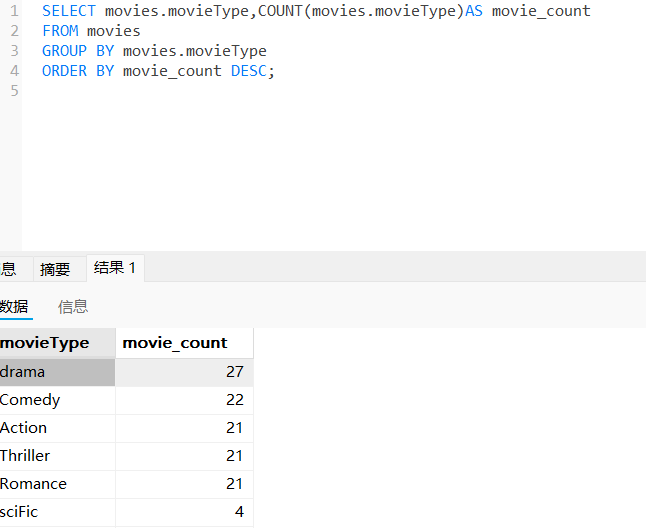
SELECT movies.movieType,COUNT(movies.movieType)AS movie\_count

FROM movies

GROUP BY movies.movieType

ORDER BY movie\_count DESC;

截图：



（8）查询出演电影的年代数最多的演员(出演多部同一年代的电影,只能算演了1个年代的电影);

语句：

WITH actor\_decades AS (

SELECT

starName,

FLOOR((movieYear - 1) / 10) \* 10 AS decade

FROM

starsin

),

unique\_actor\_decades AS (

SELECT

starName,

decade

FROM

actor\_decades

GROUP BY

starName, decade

),

actor\_decade\_count AS (

SELECT

starName,

COUNT(decade) AS decade\_count

FROM

unique\_actor\_decades

GROUP BY

starName

)

SELECT

starName,

decade\_count

FROM

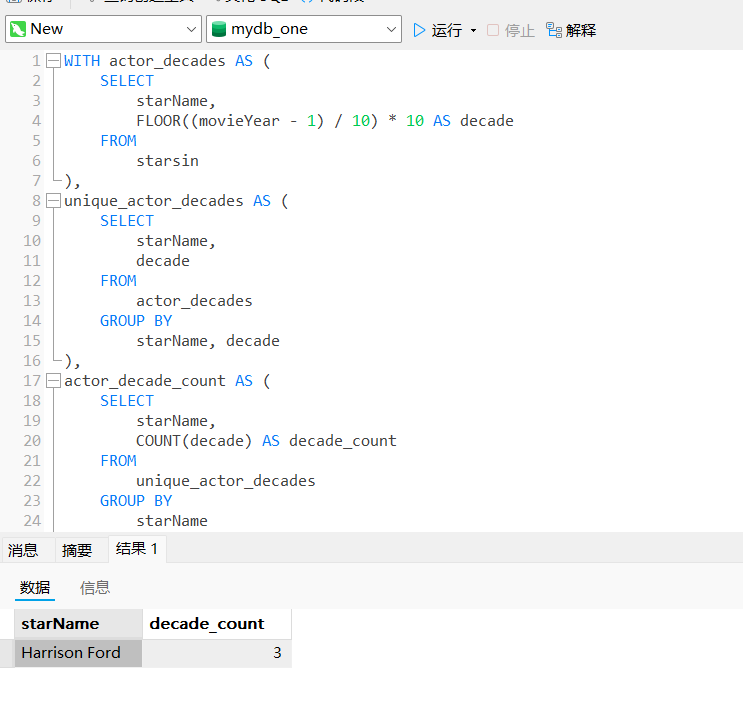
actor\_decade\_count

ORDER BY

decade\_count DESC

LIMIT 1;

截图：



（11）查询发行电影数量最多的公司名称,且公司名称长度不少于5个字符:对不同的SQL查询写法,对比一下他们的效率(提示:使用expain analize),并分析为什么会有这样的结果

语句：

使用子查询：

WITH max\_movie\_count AS (

SELECT

studioName,

COUNT(\*) AS movie\_count

FROM

movies

WHERE

LENGTH(studioName) >= 5

GROUP BY

studioName

)

SELECT

studioName,

movie\_count

FROM

max\_movie\_count

WHERE

movie\_count = (SELECT MAX(movie\_count) FROM max\_movie\_count);

使用窗口函数：

WITH studio\_movie\_count AS (

SELECT

studioName,

COUNT(\*) AS movie\_count

FROM

movies

WHERE

LENGTH(studioName) >= 5

GROUP BY

studioName

),

max\_movie\_count AS (

SELECT

MAX(movie\_count) AS max\_count

FROM

studio\_movie\_count

)

SELECT

smc.studioName,

smc.movie\_count

FROM

studio\_movie\_count smc

JOIN

max\_movie\_count mmc

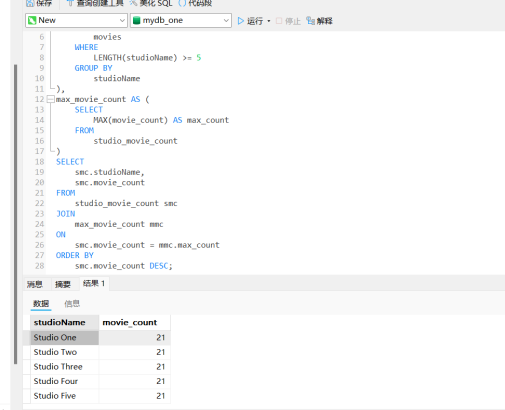
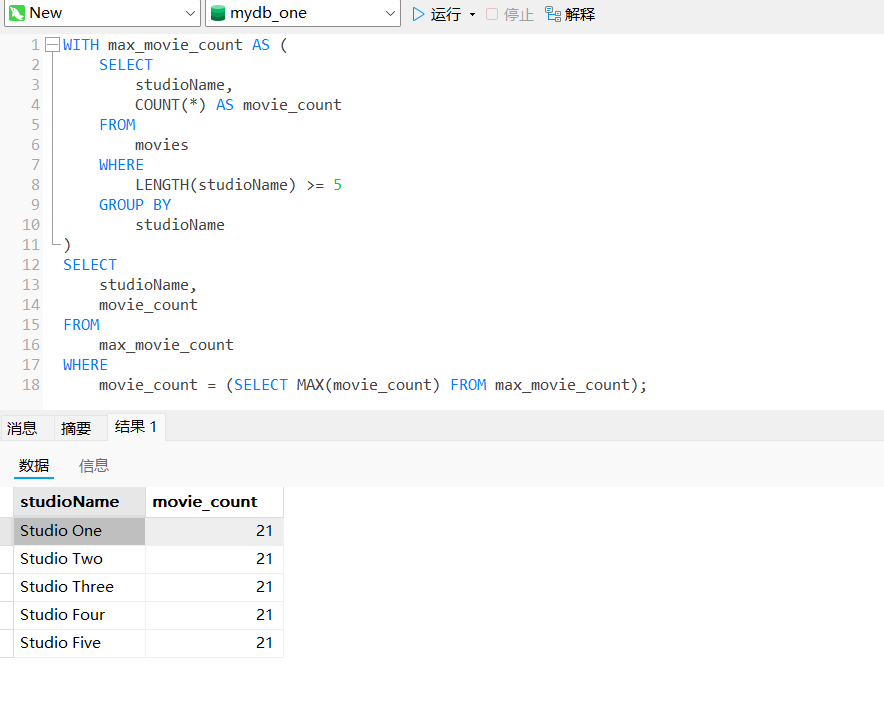
ON

smc.movie\_count = mmc.max\_count

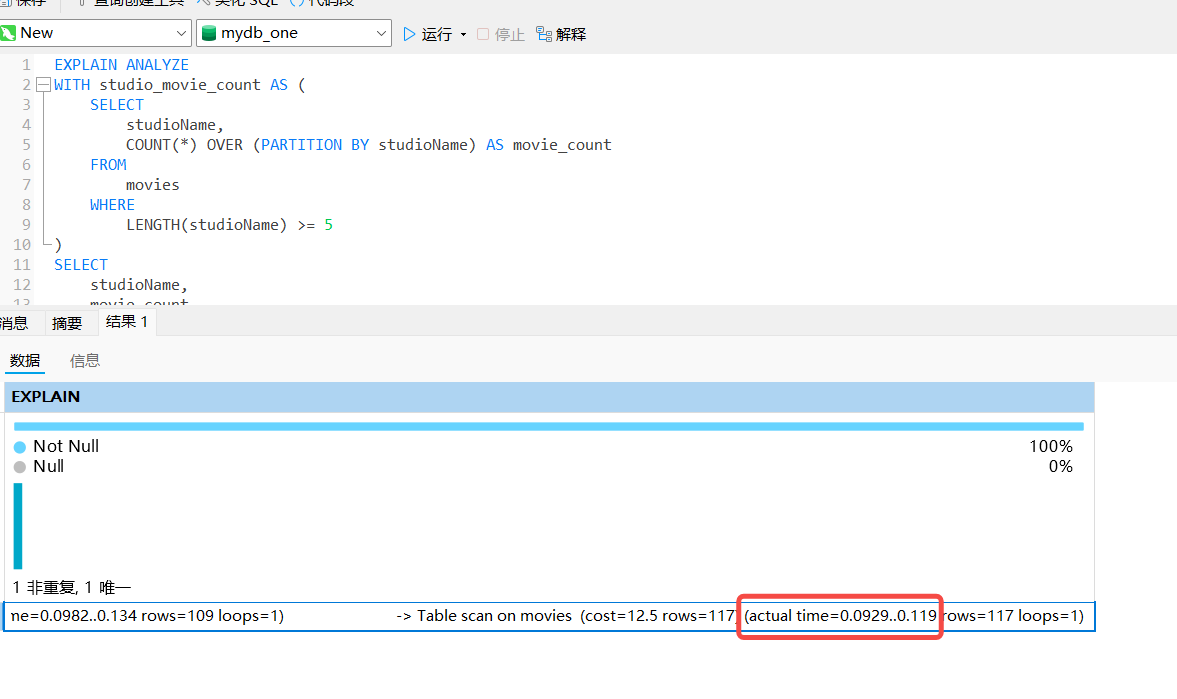
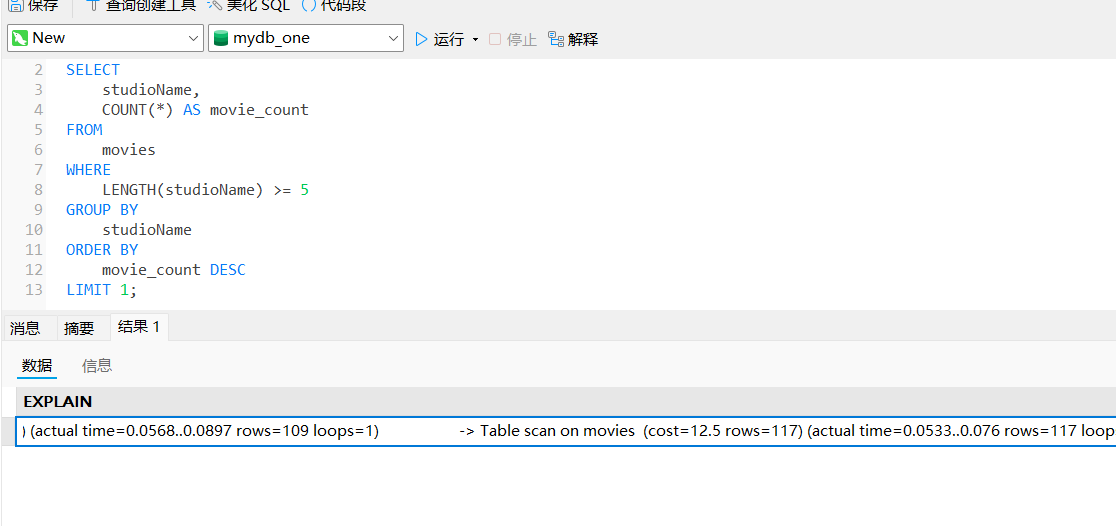
ORDER BY

smc.movie\_count DESC;

截图：



从左到右依次为子查询和窗口函数的查询结果



从左到右依次为子查询和窗口函数的速度结果

查询1：使用简单的 GROUP BY 和 ORDER BY。执行计划显示使用了哈希聚合和快速排序。

查询2：使用窗口函数。执行计划显示使用了窗口聚合和哈希聚合。

### 结论：

### 查询1 和 查询2 的执行时间相近，但 查询1 略快。这是因为 查询1 的执行计划更简单，没有额外的窗口函数开销。

总的来说，对于简单的查询，直接使用 GROUP BY 和 ORDER BY（查询1）也是很好的选择。使用窗口函数（查询2）可能会增加一些额外的开销，但在某些复杂查询中可能更有优势。

（12）查询在Harison Ford出演的电影中均扮演某个角色的演员。

语句：  
SELECT s2.starName, COUNT(DISTINCT s2.movieTitle) AS num

FROM starsin s1

JOIN starsin s2 ON s1.movieTitle = s2.movieTitle

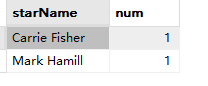
WHERE s1.starName = 'Harrison Ford'

AND s2.starName != 'Harrison Ford'

GROUP BY s2.starName

HAVING COUNT(DISTINCT s2.movieTitle) > 0;

截图：



1. 实验总结（收获及不足）

请从以下几方面来说明：

1. 对知识的掌握程度；

熟练掌握了使用SQL语言进行数据库操作的方法，包括基本的 `SELECT`、`INSERT`、`UPDATE` 和 `DELETE` 语句。

多表查询：学会了如何使用 `JOIN` 语句进行多表查询，能够处理复杂的关联关系，提取所需数据。嵌套查询：掌握了嵌套查询的使用方法，能够在主查询中引用子查询的结果。

聚集函数：熟悉了 `COUNT()`, `SUM()`, `AVG()`, `MIN()`, `MAX()` ，能够在查询中进行数据汇总和统计。

学会了如何使用CTE来简化复杂的查询逻辑，特别是在需要多次引用中间结果的情况下。

掌握了如何创建和使用临时表来存储中间结果，这对于处理复杂查询和大数据量的场景非常有用。

了解了窗口函数的基本用法，特别是 `COUNT()` 和 `MAX()` 等聚合函数在窗口中的应用，这有助于进行更灵活的数据分析。

了解了MySQL中的事务管理机制，能够使用 `BEGIN`, `COMMIT` 和 `ROLLBACK` 进行事务控制。

1. 对MySQL的熟悉程度，以及MySQL提供的相关工具或功能的熟悉程度；

进一步巩固了对MySQL查询语言的掌握，能够编写复杂的查询语句，包括多表查询、嵌套查询和聚集函数的使用。

学会了如何使用临时表和视图来存储和管理中间结果，这在处理复杂查询时非常有用。

掌握了MySQL中的事务管理机制，能够使用事务来保证数据的一致性和完整性。

1. 对交互式命令行和图形化工具的熟悉程度。

命令行工具：

练掌握了MySQL命令行客户端的使用方法。学会了使用 `mysqldump` 工具进行数据库备份。

图形化工具：

熟悉了MySQL Workbench和Navicat的界面和功能，能够进行数据库设计、查询和管理。

通过本次实验，我对SQL语言和MySQL数据库管理系统有了更深入的理解和掌握。不仅巩固了基础知识，还学到了许多新的技能和概念。在未来的实践中，我将继续提升自己的SQL水平和数据库管理能力，特别是在性能优化和高级特性方面。希望能在实际项目中更好地应用所学知识，提高工作效率和数据处理能力。