南京邮电大学

实 验 报 告

（ 2025 / 2026 学年 第 二 学期）

课程名称	数据库系统基础
实验名称	实验二：MySQL进阶开发
实验时间	2026 年 6 月 16 日
指导单位	计算机学院、软件学院、网络空间安全学院
指导教师	彭瑶

学生姓名		班级学号	
学院(系)	计算机学院	专 业	软件工程

实 验 报 告

实验名称	实验二：MySQL进阶开发		
实验类型	验证	实验学时	2

一. 实验目的和要求

（1）理解表记录的查询过程，并熟练掌握使用Navicat for MySQL进行表记录的查询操作，包括单表记录的查询、聚合函数的查询、多表连接查询以及子查询等；

（2）理解索引的概念及其作用，掌握索引的查看、创建、使用和删除方法；

（3）理解视图的概念及其作用，掌握视图的创建、查看、修改、删除及应用方法。

二、实验环境(实验设备)

硬件: 微型计算机

软件: Windows 操作系统、MySQL 5.6或更高版本、Navicat for MySQL 15或更高版本

实 验 报 告

三、实验原理及内容

1表记录的查询

在数据库studentinfo中创建一个名为"Student1"的数据表，包括字段（id，name，age，grade），并插入一些示例数据，包括但不限于：(1, 'Alice', 18, 'A'), (2, 'Bob', 20, 'B'), (3, 'Charlie', 17, 'A'), (4, 'David', 19, 'B'), (5, 'Emily', 21, 'A')；

（1）查询 Student1 表中姓名以字母 D 开头的学生信息，给出SQL语句并输出截图：

```
SELECT * FROM Student1 WHERE name LIKE 'D%';
```

查询 - studentinfo

```
1  SELECT * FROM Student1 WHERE name LIKE 'D%';
```

	id	name	age	grade
1	4	David	19	B

1行

（2）请在Student1表中找出年级为"B"且出生于2005年（含）以后的学生记录，给出SQL语句并输出截图：

```
SELECT * FROM Student1WHERE grade = 'B' AND (YEAR(CURDATE()) - age) >= 2005;
```

查询 - studentinfo

```
1  SELECT * FROM Student1
2  WHERE grade = 'B' AND (YEAR(CURDATE()) - age) >= 2005;
```

	id	name	age	grade
1	2	Bob	20	B
2	4	David	19	B

2行

（3）查询年级为A或年龄小于19岁的学生，并按年级升序、年龄降序排列，给出SQL语句并输出截图：

```
SELECT * FROM Student1WHERE grade = 'A' OR age < 19ORDER BY grade ASC, age DESC;
```

查询 - studentinfo

```
1  SELECT * FROM Student1
2  WHERE grade = 'A' OR age < 19
3  ORDER BY grade ASC, age DESC;
```

	id	name	age	grade
1	5	Emily	21	A
2	1	Alice	18	A

实 验 报 告

四、实验小结（包括问题和解决方法、心得体会、意见与建议等）

一. 实验中遇到的主要问题及解决方法

问题一：做第（2）题时，最初用的写法是 WHERE grade='B' AND age<=21，看起来结果是对的，但回头想了想这是把当前年份硬编码进去了——以后跑就错了。改成 (YEAR(CURDATE())-age)>=2005 之后才算把题目里的「出生于2005年以后」翻译准确，年份自动跟着当前时间走。

问题二：第（6）题做多表连接的时候，一开始用的是 INNER JOIN，结果 Emily 直接没出现，但题目里强调了「包括没有成绩的学生」。意识到要用 LEFT JOIN 之后又遇到了 total 列全是 NULL 的问题——只要参与运算的列里有一个是 NULL，加出来就是 NULL。用 IFNULL(g.math_grade,0)+IFNULL(g.science_grade,0) 包了一下才正常。

问题三：第（8）题的「大于所有年级平均年龄」一开始理解错了，写成 age > (SELECT AVG(age) FROM Student1)，相当于全表平均。重读了一下题，「所有年级」是每个年级算一个平均值然后都要大于，对应的就是 > ALL(...) 配 GROUP BY grade 的子查询。

问题四：做索引那一题时，SHOW INDEX FROM Student2 一开始没结果，查了下发现 idx_name 根本没建——题目默认它存在，但其实需要自己先 CREATE UNIQUE INDEX idx_name ON Student2(name)。另外组合唯一索引 (department, age) 加上去之后，(Science,20) 这种组合也确实只能存一份，顺便验证了一下唯一性约束确实生效。

（二）实验心得

这次实验主要在练 SELECT 的表达力。单表查询本身不难，卡人的往往是题目用的中文描述和 SQL 怎么对得上——比如「出生于 2005 年以后」到底是按年龄字段反推还是另加列，「大于所有年级的平均年龄」到底是大于总平均还是逐个年级都要大于。这种翻译过程其实就是把模糊需求拆成 WHERE / GROUP BY / HAVING / 子查询的过程，感觉跟平时写业务逻辑挺像的。

多表连接里最容易踩坑的是 NULL 的传染性，IFNULL/COALESCE 之类的函数之前看书的时候没太在意，这次实打实地被 total 全 NULL 的结果教了一遍。另外 LEFT JOIN 和 INNER JOIN 在「保留全部学生」这种语义里完全不能混用，这也是看文档时候不容易感知到、动手才会清楚的差别。

索引和视图的部分更偏「设计」一点。索引这次只是建和删，实际选哪个列做索引、要不要建组合索引，应该和查询模式一起考虑，目前只能算理解了机制；视图则更像把常用的复合查询起个名字，对部门汇总这种场景很合适，ALTER 用 CREATE OR REPLACE 比单纯改定义更不容易出错。整体下来对 MySQL 的「能做什么」清晰了不少，下次写后端项目应该能更自如地把查询能力下沉到 SQL 里，而不是全靠应用层 for 循环硬拼。

（三）意见与建议（没有可省略）

题目（1.6）的 Grades 表里 student_id=6 在 Student1 里其实并不存在，做 LEFT JOIN 时这

实 验 报 告

五.支撑毕业要求指标点

☑ 4.2-M能够根据实验方案，配置实验环境、开展实验，综合分析实验结果以获得合理有效的结论。

☑ 5.2-M 能够针对计算机及应用领域中的复杂工程问题，合理选择使用恰当的技术、资源和现代工程工具进行预测和模拟，并理解其局限性。

六、指导教师评语

评分细则	评分项	优秀	良好	中等	合格	不合格
	遵守实验室规章制度					
	学习态度					
	算法思想准备情况					
	程序设计能力					
	解决问题能力					
	算法设计合理性					
	算法效能评价					
	报告书写认真程度					
	内容详实程度					
	文字表达熟练程度					
	其它评价意见					
	本次实验能力达成评价（总成绩）		批阅人		日期	