

数据库系统课程实验报告

|  |  |
| --- | --- |
| 实验名称： | 数据高级查询 |
| 实验日期： | 4.11 |
| 实验地点： | 西部片区4号楼 |
| 提交日期： | 4.25 |
|  | |
| 学号： | 37220232203786 |
| 姓名： | 潘腾凯 |
| 专业年级： | 软工2023级 |
| 学年学期： | 2024-2025学年第二学期 |

1. 实验目的

* 掌握数据库的特点（字体：华文仿宋，字号：四号，下同）
* 熟练掌握设计正确查询语句以实现数据高级查询的方法
* 熟练掌握 mysql 连接查询、子查询和集合查询的方法

─ (内)连接、 (全)外连接、左外连接、右外连接

─ 子查询(嵌套查询)

─ 不相关子查询与相关子查询

─ EXISTS/NOT EXISTS 的使用

─ ANY 的使用

─ ALL 的使用

─ 集合运算： UNION【ALL】， INSERSECT， EXCEPT

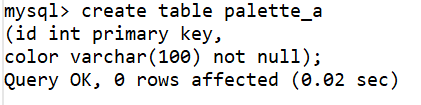
* 理解不相关子查询与相关子查询的各自特点
* 熟练掌握基于派生表的查询构造方法
* 建议： 尽量使用不同的查询语句实现同一查询要求。如， 所有带 IN 谓词、比较运算符、 ANY 或 ALL 谓词的子查询都能用带 EXISTS 谓词的子查询等价替换

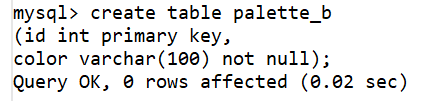
1. 实验内容和步骤

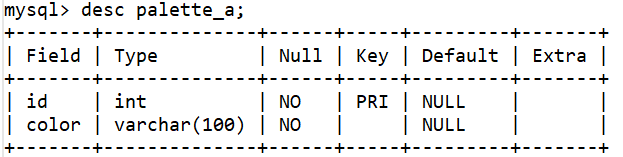
 连接运算

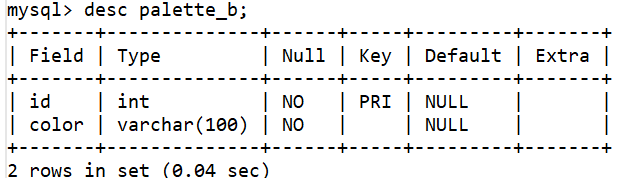
1. 创建两种表： palette\_a 和 palette\_b，它们有相同的结构

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 说明 |
| id | INT | PRIMARY KEY |
| color | VARCHAR(100) | NOT NULL |

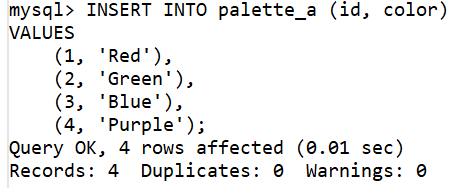


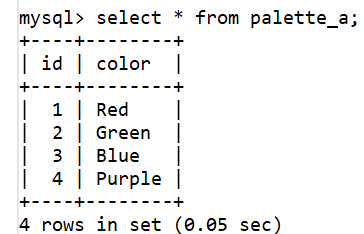




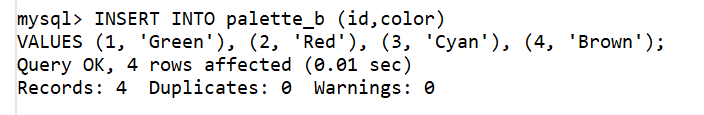


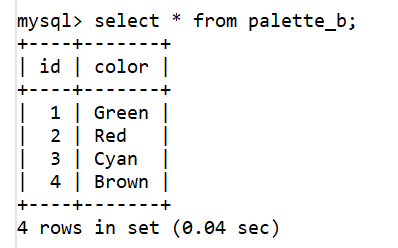
2. 为表 palette\_a 添加样例数据： {(1, 'Red'), (2, 'Green'), (3, 'Blue'), (4, 'Purple')}



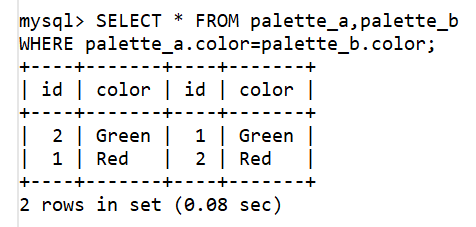


3. 为表 palette\_b 添加样例数据： {(1, 'Green'), (2, 'Red'), (3, 'Cyan'), (4, 'Brown')}

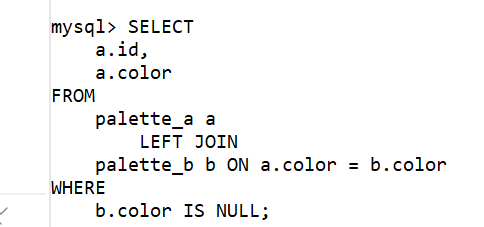


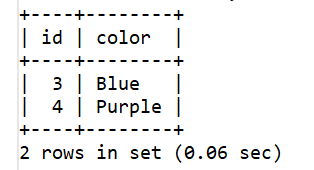


4. 查询两张表中相同颜色的所有信息

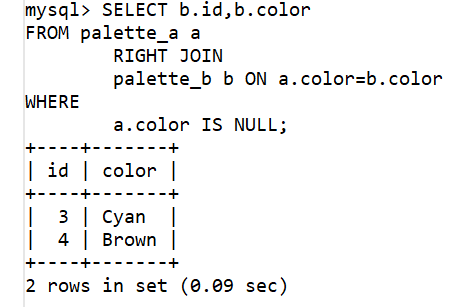


5. 查询 palette\_a 表中颜色不出现在 palette\_b 表中的 id 和颜色（要求使用左外连接实现）

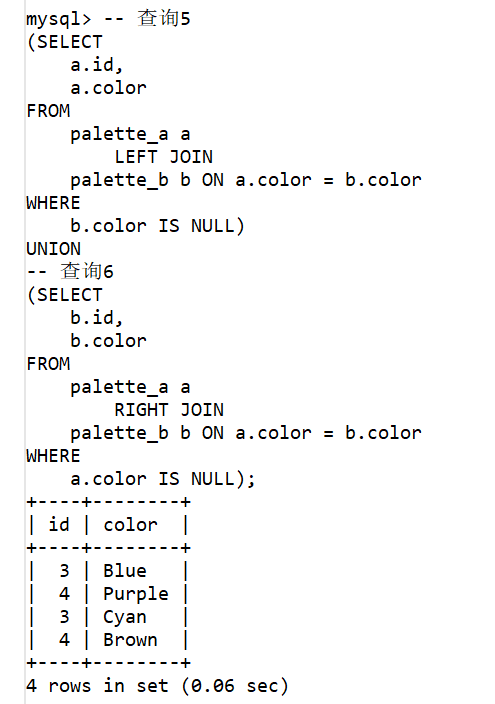




6. 查询 palette\_b 表中颜色不出现在 palette\_a 表中的 id 和颜色（要求使用右外连接实现）

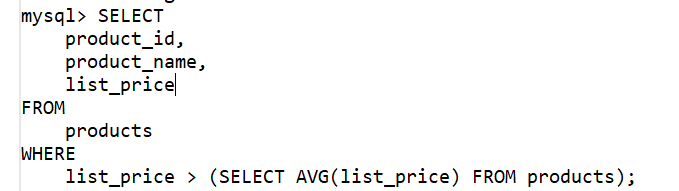


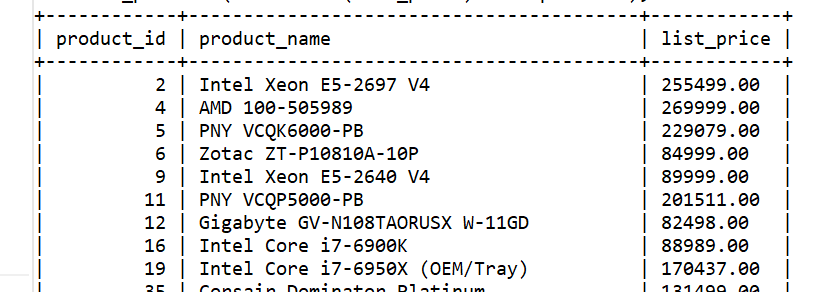
7. 查询 5 和 6 的并集（注意： mysql8 不支持 full outer join， 但可以通过 5 UNION 6 实现）

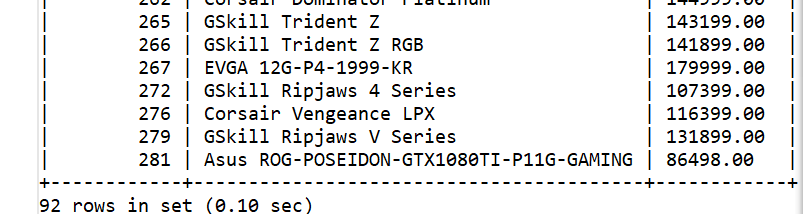


 子查询

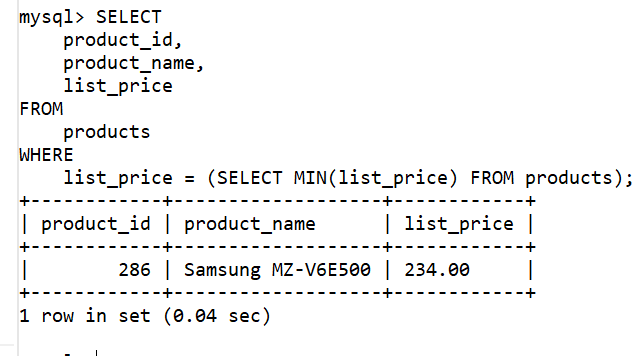
8. 查询产品表 products 中的 product\_id, product\_name, list\_price 信息，要求产品定价 list\_price 大于其平均定价 list\_price（要求使用子查询实现）





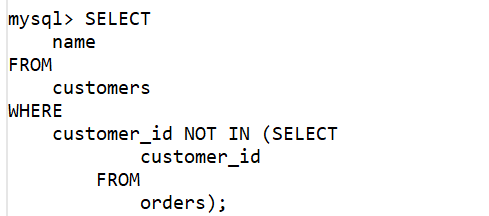


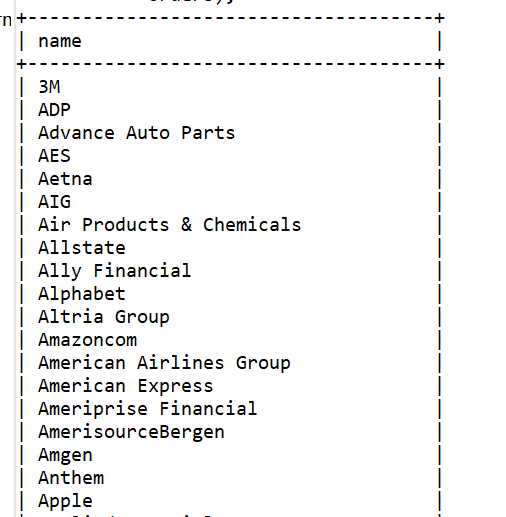
9. 查询产品表 products 中最便宜产品的 product\_id, product\_name, list\_price（要求使用子查询实现）

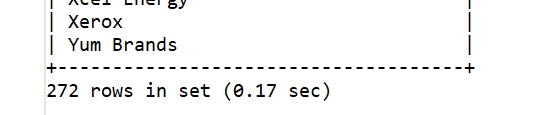


10. 查询没有一个订单的顾客姓名（要求使用使用 NOT IN 实现+其他方法（如果能找到， 鼓励， 下同））

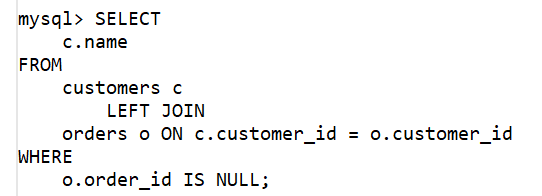
方法一：用NOT IN实现：

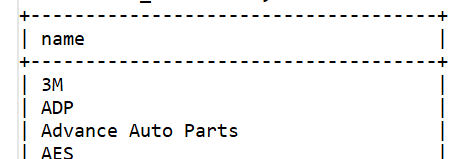


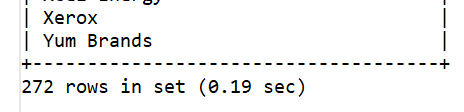




方法二：用左连接实现

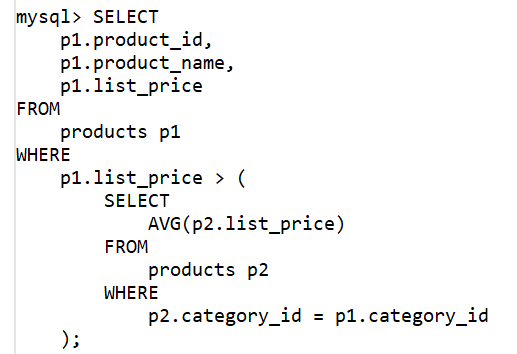


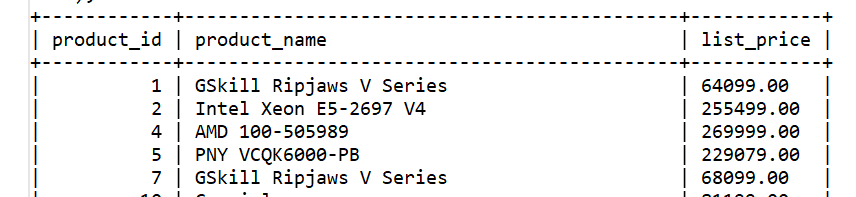


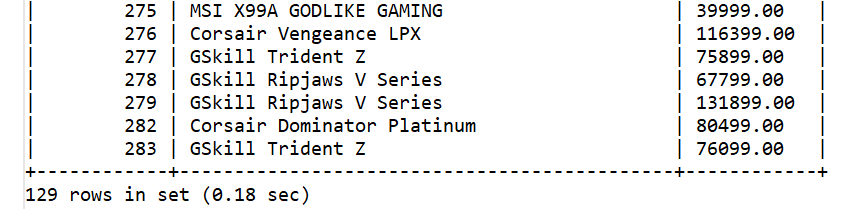


11. 查询产品表 products 中产品的 product\_id, product\_name, list\_price，要求产品定价 list\_price大于其同类产品（可由 category\_id 表达）的平均定价（要求使用相关子查询和基于派生表查询的两种方法实现）

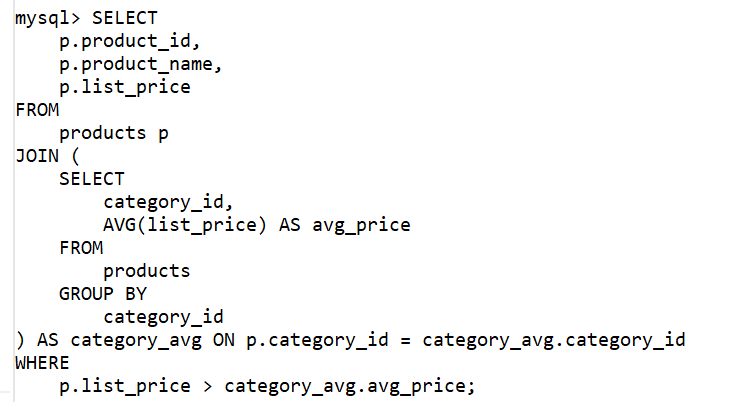
方法一：相关子查询

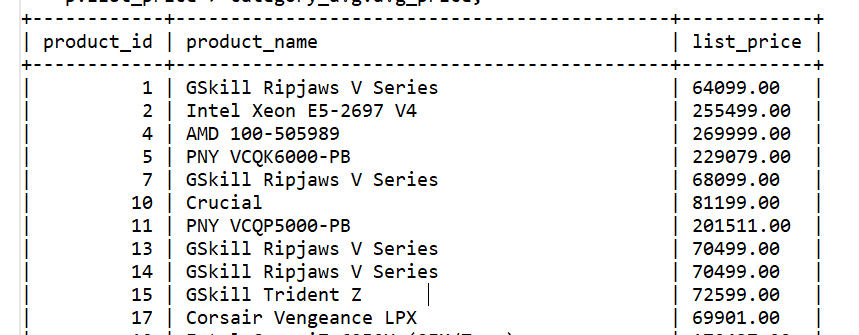


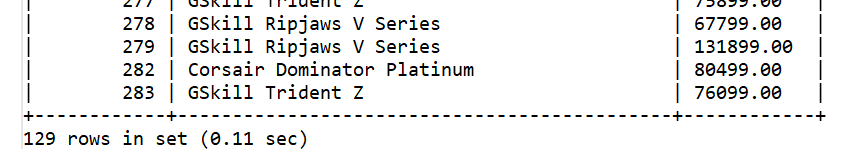




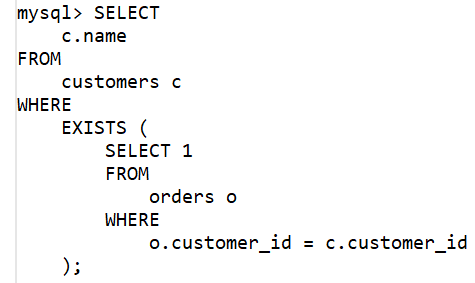
方法二：派生表查询

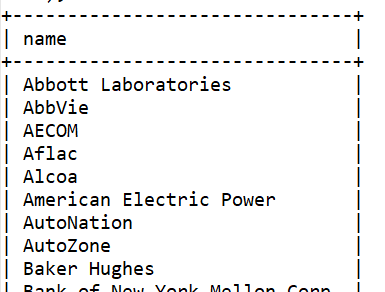


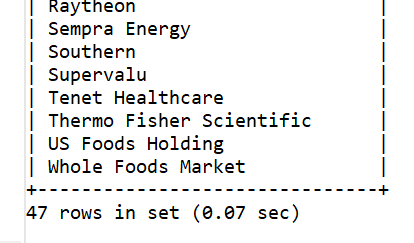




12. 查询有订单 order 的所有顾客 customer 姓名（查询涉及 customers 表和 orders 表）（要求使用EXISTS 实现）

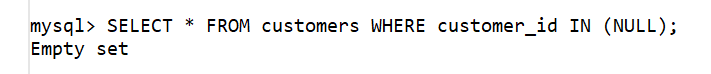






13. 执行以下三条与 NULL 相关的语句，观察各自执行的结果，能否从中得出某些初步结论？ （EXISTS 与IN 的不同）

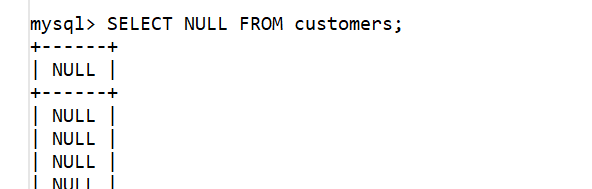
（1）SELECT \* FROM customers WHERE customer\_id IN (NULL);

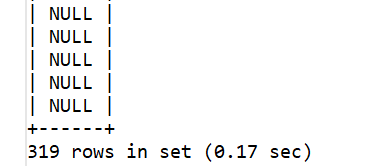


执行结果如上：没有返回任何行

结论：IN 操作符在判断时，如果列表中有 NULL 值，它遵循 SQL 的三值逻辑（TRUE、FALSE、UNKNOWN）。当 customer\_id 与 NULL 进行比较时，结果是 UNKNOWN 而不是 TRUE 或 FALSE。WHERE 子句只返回条件结果为 TRUE 的行，所以该查询不会返回任何行。

（2）SELECT NULL FROM customers;

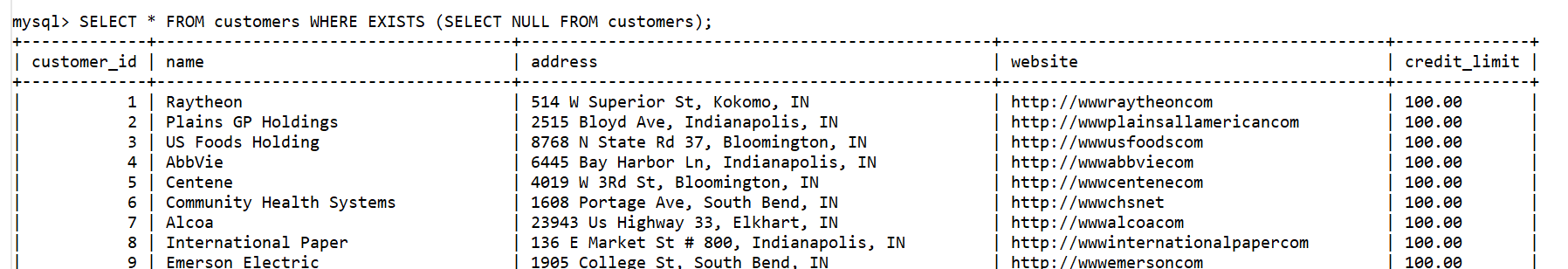


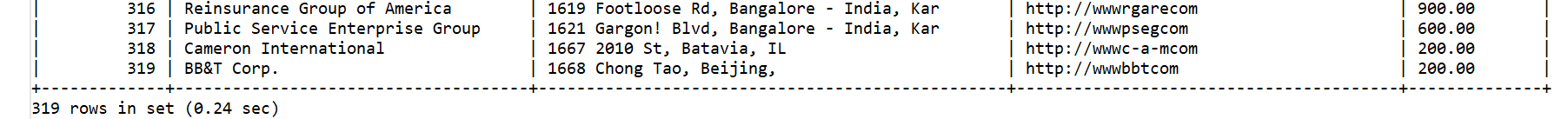


执行结果如上：返回与 customers 表行数相同数量的结果，每一行只有一个值 NULL。

结论： 此查询实际上是从customers 表中选取数据，不过选取的列固定为 NULL。因此，对于 customers 表中的每一行，都会生成一个包含 NULL 的结果行。

（3）SELECT \* FROM customers WHERE EXISTS (SELECT NULL FROM customers);

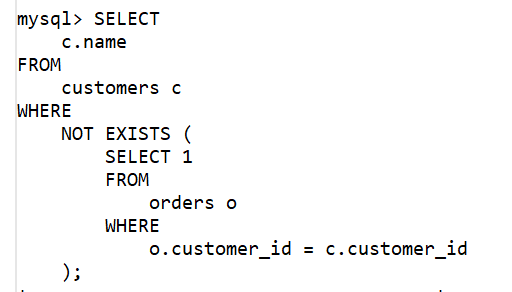


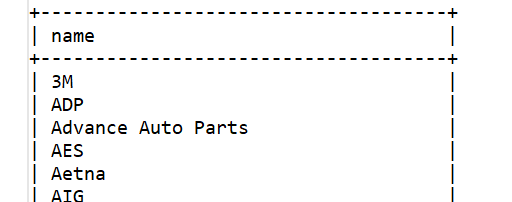


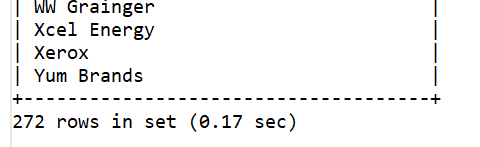
执行结果如上：返回customers表中的所有行

结论：EXISTS 操作符只关心子查询是否返回至少一行数据，而不关心子查询返回的具体内容。SELECT NULL FROM customers 会返回与 customers 表行数相同数量的行，所以子查询至少会返回一行数据，这使得 EXISTS 条件为 TRUE，主查询会返回 customers 表中的所有行。

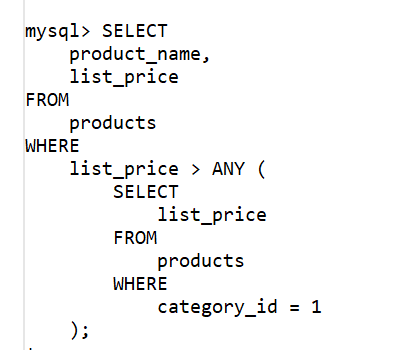
1. 找出所有没有订单的顾客姓名（查询涉及 customers 表和 orders 表）（要求使用 NOT EXISTS 实现）

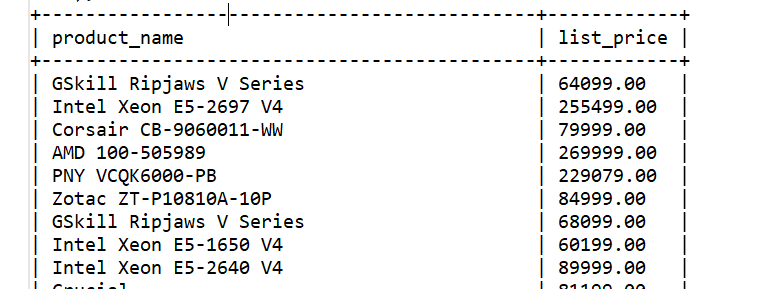


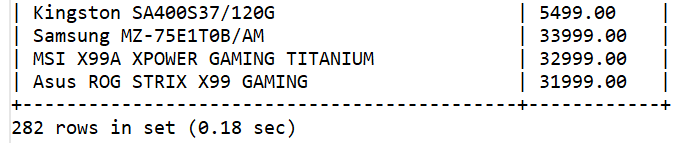




1. 查询产品表 products 中的产品名 product\_name 和定价 list\_price，要求其定价高于产品种类 1 中的任何产品定价（要求使用 ANY 实现）

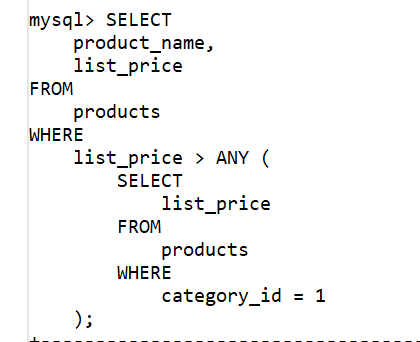


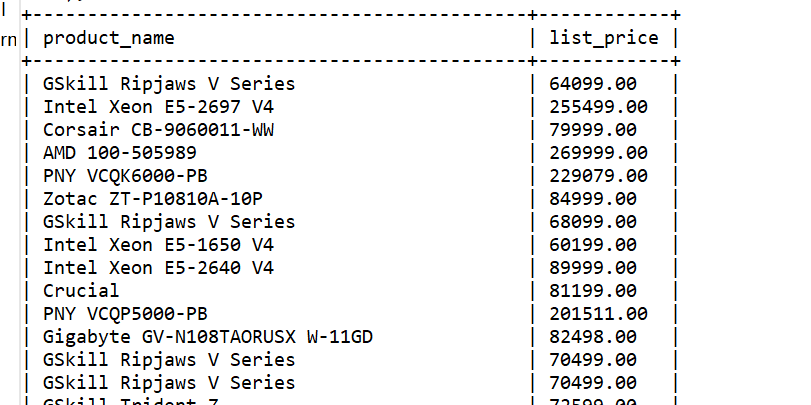


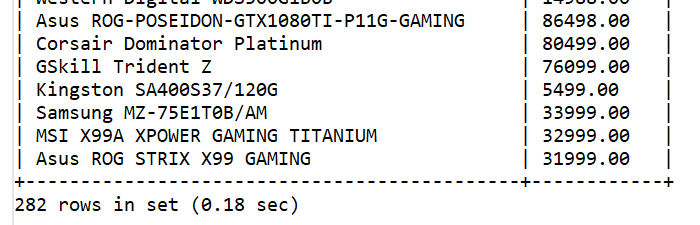


16. 查询产品表 products 中的产品名 product\_name 和定价 list\_price，要求其定价高于产品种类 1 中

的所有定价（要求使用 ALL 实现）

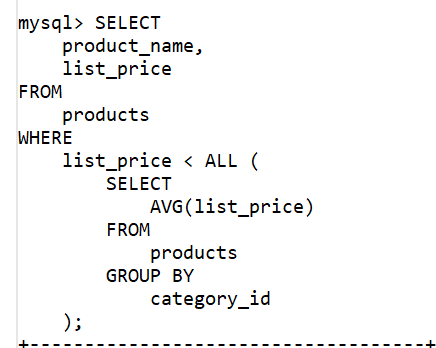


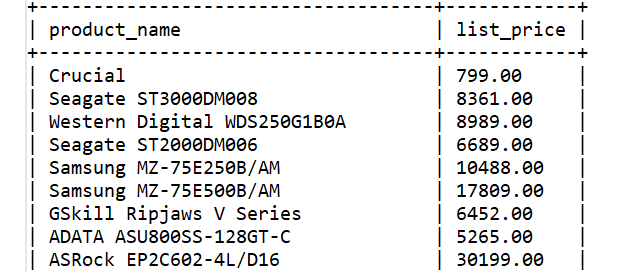


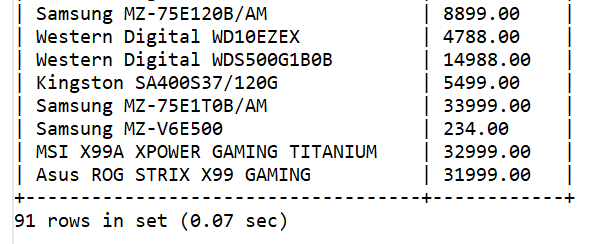


17. 查询产品表 products 中的产品名 product\_name 和定价 list\_price，要求其定价低于产品种类的所

有平均定价（要求使用 ALL 实现）



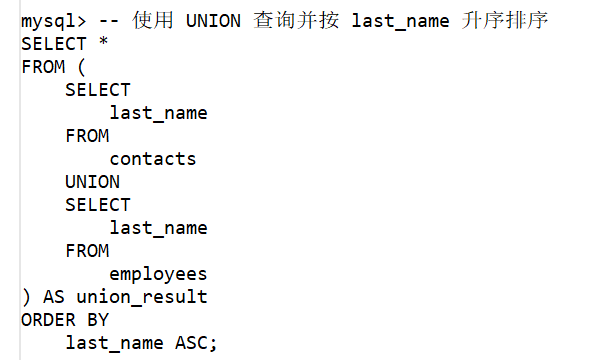


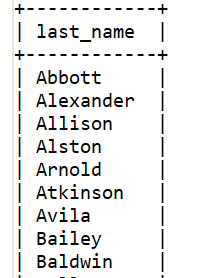


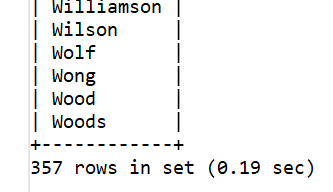
 集合查询

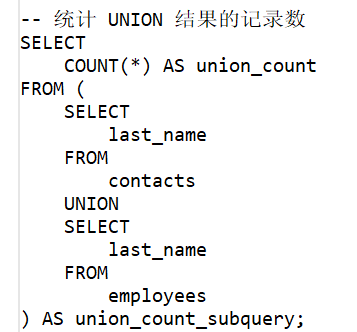
1. 查询 contacts 表和 employees 表中的所有 last\_name，并以 last\_name 升序显示（要求分别使用UNION 和 UNION ALL 实现，统计各自结果的记录数，比较两者数目是否相同，为什么？ ）

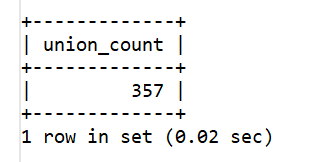
UNION：



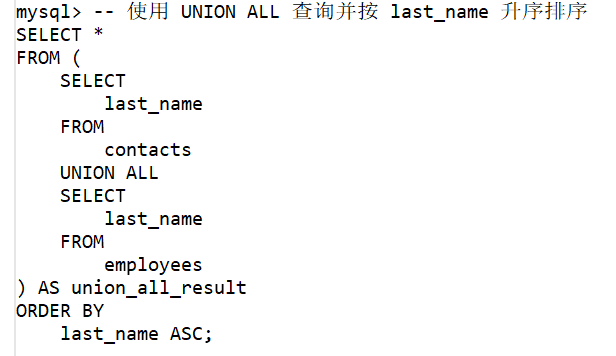


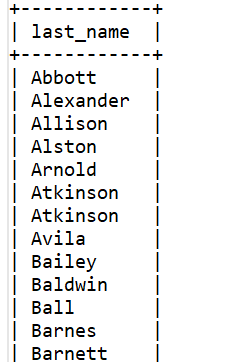


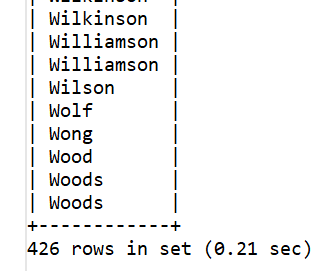


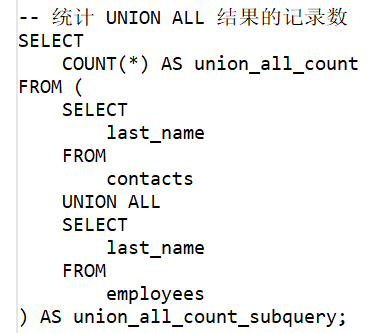


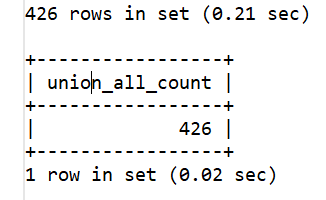
UNION ALL：











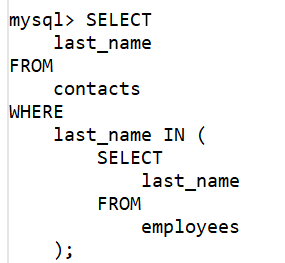
可以看出，UNION ALL 结果的记录数大于 UNION 结果的记录数。

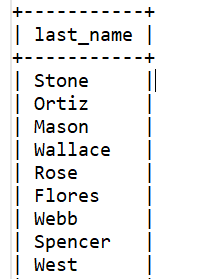
原因：如果 contacts 表和 employees 表中的 last\_name 没有重复值，那么 UNION 和 UNION ALL 结果的记录数会相同。

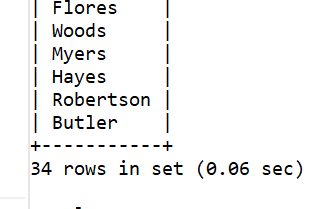
如果存在重复的 last\_name，UNION 会去除这些重复记录，而 UNION ALL 会保留所有记录，所以 UNION ALL 的记录数会多于 UNION 的记录数。

19. 查询同时出现在 contacts 表和 employees 表中的所有 last\_name（要求使用 INERSECT+其他方法实现（如果能找到））

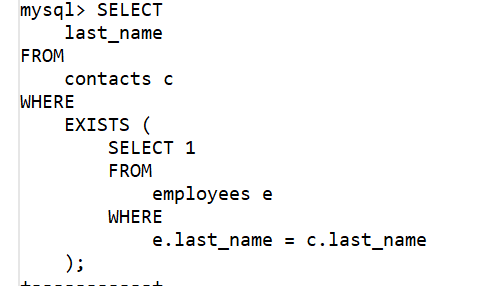
方法一：INERSECT

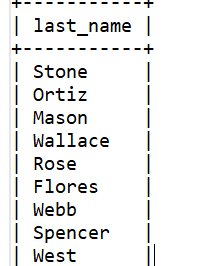


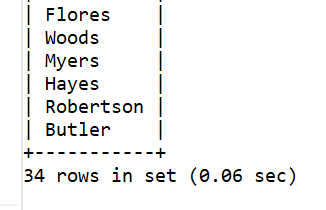




方法二：EXISTS

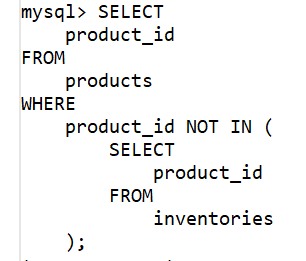


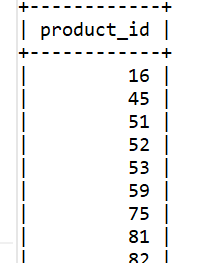


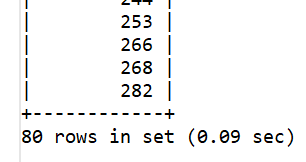


20. 查询在产品表 products 中而不在库存表 inventories 中的产品号 product\_id（要求使用 EXCEPT+其他方法实现（如果能找到））

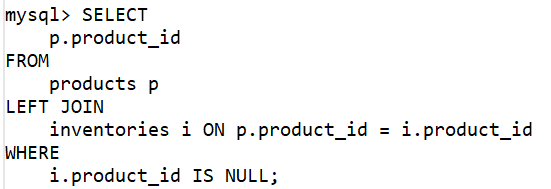
方法一：

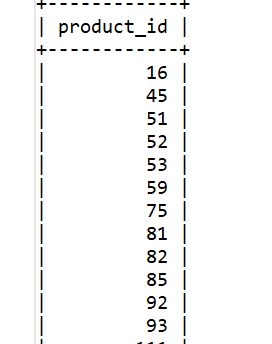


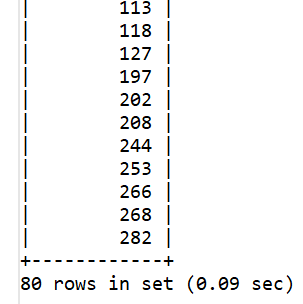




方法二： LEFT JOIN







1. 实验总结

3.1 完成的工作

完成了一些高级查询操作，实践操作并应用了UNION、JOIN、EXISTS、IN、子查询、集合查询等功能。

3.2 对实验的认识

对实验的思考（思考题）：

1. 请分析子查询的特点及归纳总结什么场景下的查询必须使用子查询才能实现， 举例说明之。

子查询的特点

**嵌套性**：子查询是嵌套在主查询中的查询语句，它可以出现在主查询的 SELECT、FROM、WHERE、HAVING 等子句中。例如，在 WHERE 子句中使用子查询来筛选满足特定条件的记录。

**独立性**：子查询本身是一个独立的查询，它先于主查询执行，并且可以独立返回结果。子查询的结果通常用于主查询的条件判断、数据过滤或作为主查询的数据源。

**灵活性**：子查询可以使用各种查询语句和操作符，包括 SELECT、JOIN、GROUP BY、ORDER BY 等，能够根据复杂的业务需求进行灵活的组合和嵌套，以实现复杂的查询逻辑。

必须使用子查询的场景及举例

**查询依赖于其他查询结果的场景**：当查询条件依赖于另一个查询的结果时，必须使用子查询。例如，要查询每个部门中工资高于该部门平均工资的员工信息。首先需要通过子查询计算出每个部门的平均工资，然后在主查询中使用这个结果来筛选员工。

**获取不存在于其他表中的数据场景**：比如要查询没有订单的客户信息，就需要使用子查询来判断哪些客户在订单表中没有对应的订单记录。

2.分析总结相关子查询与不相关子查询的区别， 试举例说明使用相关子查询的典型场景， 如何将相关子查询转换为一般查询实现？（说明：一般查询是指不一定必须使用子查询的查询）

**区别：**

执行次数：

不相关子查询：不相关子查询只执行一次，它独立于主查询，子查询的结果被主查询引用，就像使用一个常量或一个固定的数据集一样。例如，查询所有工资高于公司平均工资的员工，子查询 SELECT AVG(salary) FROM employees 只执行一次来计算出公司平均工资，然后主查询使用这个结果进行筛选。

相关子查询：相关子查询会为外层查询的每一行执行一次。因为子查询中引用了外层查询的表或列，所以每次外层查询的行发生变化时，子查询都需要重新计算。例如，查询每个部门中工资最高的员工，子查询需要为每个部门分别找出最高工资，即对于 employees 表中的每一行，都要根据其 department\_id 在子查询中计算该部门的最高工资。

依赖关系：

不相关子查询：子查询与主查询之间没有直接的依赖关系，子查询可以单独执行并返回结果，其结果不依赖于主查询中的任何值。

相关子查询：子查询依赖于主查询中的列值，子查询的执行需要根据主查询当前行的相关值来确定。例如在上述查询每个部门中工资最高的员工的例子中，子查询 SELECT MAX(salary) FROM employees sub WHERE sub.department\_id = e.department\_id 中的 sub.department\_id = e.department\_id 建立了子查询与主查询中 employees 表当前行的 department\_id 的关联。

**使用相关子查询的典型场景及转换为一般查询的方法：**

典型场景：除了上述提到的查询每个部门中工资最高的员工这类场景外，还有如查询每个客户最近一次订单的信息等。

转换为一般查询的方法：以查询每个部门中工资最高的员工为例，可以通过使用 JOIN 和 GROUP BY 来实现。

3.3 遇到的困难及解决方法

在步骤18，遇到 1064 - You have an error in your SQL syntax 错误时，通过仔细检查报错信息中指出的错误位置附近的代码，发现是由于在使用 UNION 操作符时，对结果集进行排序的语法错误。经过修正后，成功解决了问题，这让我明白了在编写 SQL 语句时，要严格遵循语法规则，并且对一些复杂的操作要仔细检查其使用方法。