第5章 在线销售助农产品系统的数据查询

数据库通常包含大量数据，要从海量的数据中找到需要的某条记录无异于大海捞针。

通过本章的学习，读者不仅能够掌握如何利用 SQL 等查询语言高效地与数据库对话，提取所需信息，还能学会如何在复杂多变的业务场景中，灵活运用数据查询技术解决实际问题，优化农产品销售策略，提升用户体验，最终助力农业电商平台的稳健发展与农产品的畅销无阻。

【知识目标】

- 能够认识到数据库作为系统核心组成部分的重要性，掌握 SELECT 语句的基本使用，包括指定查询的列和表，这是从数据库中提取数据的基础。
- 熟悉使用 WHERE 子句设置查询条件，实现数据的精确筛选；同时，理解并应用 ORDER BY 子句对数据进行排序，确保结果按指定顺序展示。
- 了解并掌握聚合函数 SUM()、AVG()、COUNT()等的使用，对数据进行统计分析；通过 GROUP BY 子句对数据进行分组，以执行更复杂的聚合分析。
- 能够深入理解并掌握连接查询 INNER JOIN、LEFT JOIN 等的概念与应用，能够跨多个表查询相关数据，整合逻辑上相关的数据集合。
- 探索并掌握子查询和存在量词等高级查询技巧，以执行更复杂的查询逻辑，如多层嵌套查询、存在性测试等，进一步提升数据查询与分析的能力。

【能力目标】

能够培养读者的数据查询与分析能力，包括精准查询构建、复杂数据处理、统计分析与报告生成，以及解决实际数据问题的逻辑思维与创新能力。

【素质目标】

在数据查询过程中，读者将学习如何解决实际数据问题。培养读者数据敏感度、逻辑思维严谨性、问题解决能力及持续学习新技术的能力，使其成为高效的数据分析师或数据库管理者。

5.1 简单查询

数据库存在的意义在于将数据组织在一起，以方便查询。“查询”的含义就是用来描述从数据库中获取数据和操纵数据的过程。SQL 中最主要、最核心的部分是它的查询功能。查询语言用来对已经存在于数据库中的数据按照特定的组合、条件表达式或者一定次序进行检查。SQL 使用 SELECT 语句来实现数据的查询，并按用户要求查询数据，将查询结果以表格的形式返回。

1. SELECT 语句的精简语法

SELECT 查询语句功能强大，语法较为复杂，下面介绍 SELECT 语句精简结构的语法：

```
SELECT  SELECT_LIST
[INTO NEW_TABLE_NAME]
FROM  TABLE_LIST
[WHERE SEARCH_CONDITIONS]
[GROUP BY GROUP_BY_LIST]
[HAVING SEARCH_CONDITIONS]
[ORDER BY ORDER_LIST [ASC | DESC ] ]
```

说明：SELECT 指定了要查看的列（字段），FROM 指定了这些数据来自哪里（表或视图）。

参数说明如下。

SELECT_LIST：表示需要查询的字段的列表，字段名称之间使用逗号分隔。在这个列表中不但可以包含数据表或视图中的字段名称，还可以包含其他表达式，如常量或 SQL 函数。如果使用 * 来代替字段的列表，那么系统将返回数据表中的所有字段。

INTO NEW_TABLE_NAME：该子句将指定使用查询出来的结果集创建一个新的数据表。NEW_TABLE_NAME 为这个新数据表的名称。

FROM TABLE_LIST：使用这个句子指定查询数据的数据表的列表。

GROUP BY GROUP_BY_LIST：GROUP BY 子句根据参数 GROUP_BY_LIST 提供的字段将结果集分成组。

HAVING SEARCH_CONDITIONS：HAVING 子句是应用于结果集的附加筛选，SEARCH_CONDITIONS 用来定义筛选条件。从逻辑上讲，HAVING 子句从中间结果集对记录进行筛选，这些中间结果集是用 SELECT 语句中的 FROM、WHERE 或 GROUP BY 子句创建的。

ORDER BY ORDER_LIST [ASC | DESC]：ORDER BY 子句用来定义结果集中的记录排列的顺序。ORDER_LIST 指定排序时需要依据的字段的列表，字段之间使用逗号分隔。ASC 和 DESC 关键字分别指定记录是按升序还是降序排序。

2. SELECT 语句的执行过程

（1）读取 FROM 子句中基本表、视图的数据，执行笛卡儿积操作。

（2）选取满足 WHERE 子句中给出的条件表达式的元组。

（3）按 GROUP BY 子句中指定列的值分组，同时提取满足 HAVING 子句中组条件表达式的那些组。

（4）按 SELECT 子句中给出的列名或列表达式求值输出。

（5）ORDER BY 子句对输出的目标进行排序，按附加说明 ASC 升序排列，或 DESC 降序排列。

简单的 TRANSACT-SQL 查询只包括选择列表、FROM 子句和 WHERE 子句。它们分别说明所查询列、查询的表或视图，以及搜索条件等。

3. 选择列表

选择列表（SELECT_LIST）指出所查询的列，它可以由一组列名列表、星号、表达式、变量（包括局部变量和全局变量）等构成。

1）查询表中所有的列

```
SELECT * FROM  TABLE_NAME;
```

【例 5.1】 查询 AProductInfo 表中的所有数据。

```
SELECT *
FROM  AProductInfo;
```

执行以上代码的查询结果如图 5.1 所示。

AID	AName	OutPrice	InPrice	Category	Inventory
1	红富士	12.00	8.00	水果	100
2	白牛奶葡萄	10.00	7.00	水果	80
3	杏花鸡	26.00	20.00	禽畜	90
4	宁夏菜心	5.00	3.00	蔬菜	68
5	土豆	3.00	1.50	蔬菜	75
6	腊牛肉	60.00	50.00	禽畜	46
7	蛇果	13.00	9.00	水果	87
8	黑土猪	30.00	20.00	禽畜	30
9	西兰花	6.00	3.00	蔬菜	40
10	豆角	4.00	2.50	蔬菜	50
11	山羊肉	40.00	28.00	禽畜	55

图 5.1　例 5.1 的查询结果

2）查询表中指定的列

```
SELECT  COLUMN_NAME1[,COLUMN_NAME2,...]  FROM  TABLE_NAME;
```

说明：多个字段用逗号(英文状态输入)隔开，查询结果集合中数据的排列顺序与选择列表中所指定的列名排列顺序相同。

【例 5.2】 查询 AProductInfo 表中的农产品名称、售价两列的信息。

```
SELECT  AName,OutPrice
FROM  AProductInfo;
```

执行以上代码的查询结果如图 5.2 所示。

AName	OutPrice
红富士	12.00
白牛奶葡萄	10.00
杏花鸡	26.00
宁夏菜心	5.00
土豆	3.00
腊牛肉	60.00
蛇果	13.00
黑土猪	30.00
西兰花	6.00
豆角	4.00
山羊肉	40.00

图 5.2　例 5.2 的查询结果

3）使用别名

在显示结果时，可以指定以别名来代替原来的字段名称，定义格式有两种：

```
原列名  新列标题
```

或

```
原列名 AS  新列标题
```

注意：别名可以是任何有效的标识符，包括字母、数字和下画线，但不能以数字开头。如果别名包含空格或特殊字符，或者与 MySQL 的保留字冲突，那么需要使用反引号(`)将别名括起来。如果指定的列标题不是标准的标识符格式时，应使用引号定界符。例如，下列语句使用大写字母显示列标题。

【例 5.3】 查询 AProductInfo 表中的农产品名称、进货价、售价、类别、库存量的信息，要求字段使用汉字显示列标题。

```
SELECT AName 农产品名称, InPrice AS 进货价, OutPrice AS 售价, Category 类别, Inventory 库存量
FROM  AProductInfo;
```

执行以上代码的查询结果如图 5.3 所示。

农产品名称	进货价	售价	类别	库存量
红富士	8.00	12.00	水果	100
白牛奶葡萄	7.00	10.00	水果	80
吉花鸡	20.00	26.00	禽畜	90
宁夏菜心	3.00	5.00	蔬菜	68
土豆	1.50	3.00	蔬菜	75
腊牛肉	50.00	60.00	禽畜	46
蛇果	9.00	13.00	水果	87
黑土猪	20.00	30.00	禽畜	30
西兰花	3.00	6.00	蔬菜	40
豆角	2.50	4.00	蔬菜	50
山羊肉	28.00	40.00	禽畜	55

图 5.3　例 5.3 的查询结果

4) 消除重复行

在 SELECT 语句中，如果不指定 ALL 或 DISTINCT，默认行为是 ALL。这意味着查询将返回所有符合条件的行，包括重复的行。显式地使用 ALL 选项实际上不会改变查询的结果，因为 ALL 是默认行为。例如，SELECT ALL * FROM table_name;和 SELECT * FROM table_name;是等效的。使用 DISTINCT 选项时，查询将返回所有符合条件的唯一行，这意味着如果有多行数据在指定的列(或列组合)上完全相同，则 SELECT DISTINCT 只会返回其中一行。例如，SELECT DISTINCT column1,column2 FROM table_name;将返回 column1 和 column2 组合唯一的所有行。

【例 5.4】 查询 AProductInfo 表中不重复的农产品类别记录。

```
SELECT  DISTINCT  Category
FROM  AProductInfo;
```

【例 5.5】 查询 BuyerInfo 表中购买人姓名和性别不重复的记录。

```
SELECT  DISTINCT BName, Bsex
FROM BuyerInfo;
```

5) 限制返回的行数

在数据查询时，经常需要查询最好的、最差的、最前的、最后的几条记录，这时需要使用 LIMIT 关键字进行数据查询。

限制返回的行数：使用 LIMIT 子句可以指定查询结果返回的行数。例如，SELECT *

FROM table_name LIMIT 10;会返回 table_name 表中的前 10 行数据。

结合 OFFSET 使用：LIMIT 子句还可以与 OFFSET 子句结合使用，以跳过指定数量的行后返回结果。例如，SELECT * FROM table_name LIMIT 10 OFFSET 20;会跳过前 20 行，然后返回接下来的 10 行数据。

【例 5.6】 查询 BuyerInfo 表中前 5 条记录。

```
SELECT *
FROM  BuyerInfo
LIMIT 5;
```

【例 5.7】 查询 BuyerInfo 表中跳过前 2 行，返回接下来的 3 条记录。

```
SELECT *
FROM  BuyerInfo
LIMIT 3 OFFSET 2;
```

6）复制表

在 MySQL 中，复制表格通常涉及创建一个新表，并将现有表的数据和结构复制到新表中。主要通过使用 CREATE TABLE...LIKE 语句和 INSERT INTO...SELECT 语句来实现。

方法一：复制表结构和数据。在 MySQL 中，通常使用 CREATE TABLE...LIKE 语句来创建一个与现有表结构相同的新表，但不包括数据。然后使用 INSERT INTO...SELECT 语句将现有表的数据插入新表中。例如：

```
CREATE TABLE new_table LIKE existing_table;    -- 首先创建新表(如果尚不存在)
INSERT INTO new_table (column1, column2, ...) SELECT column1, column2, ... FROM existing_table
WHERE some_condition;      -- 然后从现有表中选择数据并插入新表中
```

【例 5.8】 将 BuyerInfo 表的数据复制到一个新表 copyBuyerInfo 中。

```
CREATE TABLE copyBuyerInfo LIKE BuyerInfo;
INSERT INTO copyBuyerInfo SELECT * FROM BuyerInfo;
```

方法二：直接复制整个表(包括数据和结构)。如果直接复制整个表，包括数据、索引、触发器等，可以使用 CREATE TABLE ... AS SELECT 语句，但这种方法不会自动复制索引、约束和触发器，因此它更适合简单场景。例如：

```
CREATE TABLE new_table AS SELECT * FROM existing_table;
```

注意：这种方法不会复制表的索引、主键、外键、触发器等结构信息。

【例 5.9】 将 BuyerInfo 表的数据复制到一个新表 CpBuyerInfo 中。

```
CREATE TABLE CpBuyerInfo AS SELECT * FROM  BuyerInfo;
```

4. FROM 子句

FROM 子句指定 SELECT 语句查询及与查询相关的表或视图。在 FROM 子句中最多可指定 256 个表或视图，它们之间用逗号分隔。

在 FROM 子句同时指定多个表或视图时，如果选择列表中存在同名列，这时应使用对象名限定这些列所属的表或视图。

语法格式如下：

```
[FROM {源表名|源视图 [,...n] }]
```

在 FROM 子句中可用以下两种格式为表或视图指定别名：

表名　AS　别名

或

表名　别名

【例 5.10】 查询购买人姓名、农产品名称和购买数量。

```
SELECT B.BName, P.AName, O.Oquantity
FROM BuyerInfo AS B, OrderInfo O, AProductInfo P
WHERE B.BID = O.BID AND O.AID = P.AID;        -- 查询条件,查询字段在三张表,这三张表要有相关
-- 联的字段,条件查询见 5.2 节
```

5.2　带条件的查询

WHERE 子句指定了要查询哪些行(记录)，通过设置查询条件，过滤掉不需要的数据行。

语法格式如下：

```
WHERE <查询条件>
```

【例 5.11】 在 AProductInfo 表中，查询售价大于 5 的所有农产品信息。

```
SELECT  *
FROM  AProductInfo
WHERE  OutPrice > 5
```

WHERE 子句可包括各种条件运算符。

比较运算符(大小比较)：＞、＞＝、＝、＜、＜＝、＜＞(或 !＝)。

范围运算符(表达式值是否在指定的范围)：BETWEEN...AND...和 NOT BETWEEN...AND...

列表运算符(判断表达式是否为列表中的指定项)：IN (项 1,项 2...)和 NOT IN(项 1,项 2...)。

模式匹配符(判断值是否与指定的字符通配格式相符)：LIKE、NOT LIKE。

空值判断符(判断表达式是否为空)：IS NULL、IS NOT NULL。

逻辑运算符(用于多条件的逻辑连接)：NOT、AND、OR。

1. 比较查询条件

比较查询条件由表达式的双方和比较运算符组成，如表 5.1 所示，系统将根据该查询条件的真假来决定某一条记录是否满足该查询条件，只有满足该查询条件的记录才会出现在最终结果集中。

表 5.1　比较运算符

运　算　符	含　　义	运　算　符	含　　义
=	等于	<>或!=	不等于
>	大于	<	小于
>=	大于或等于	<=	小于或等于

【例 5.12】 查询农产品名称为山羊肉的农产品信息。

```
SELECT *
FROM  AProductInfo
WHERE AName = '山羊肉';
```

【例 5.13】 查询所有类别不是水果的农产品信息。

```
SELECT  *
FROM  AProductInfo
WHERE Category <>'水果';
```

或

```
SELECT  *
FROM  AProductInfo
WHERE Category!= '水果';
```

【例 5.14】 查询购买数量大于或等于 6 的所有订单信息。

```
SELECT  *
FROM  OrderInfo
WHERE  Oquantity > = 6;
```

2. 逻辑运算符

优先级为 NOT、AND、OR。

1）逻辑非(NOT)

用于反转查询条件的结果，即对指定的条件取反。

【例 5.15】 查询性别为女性的购买人信息，显示购买人手机号码、姓名和电子邮箱。

```
SELECT BPhone,BName,Email
FROM  BuyerInfo
WHERE NOT Bsex = '男';
```

2）逻辑与(AND)

当连接两个布尔型表达式时，若所有表达式都为 TRUE 且不为 NULL，结果为 TRUE；若一个或多个表达式为 FALSE，结果为 FALSE；表达式中有任何一个为 NULL，结果为 NULL。可以通过括号更改计算次序。

【例 5.16】 查询姓名为欧阳松松，且是女性的购买人信息。

```
SELECT *
FROM  BuyerInfo
WHERE  BName = '欧阳松松'  AND  Bsex = '女';
```

3）逻辑或(OR)

当连接两个布尔型表达式时，若两个表达式都为 NOT NULL，如果有任意一个表达式为 TRUE，则结果为 TRUE，否则结果为 FALSE；若有一个表达式为 NULL，如果另一个表

达式为 TRUE,则结果为 TRUE,否则结果为 NULL;若两个表达式均为 NULL,结果为 NULL。AND 的优先级高于 OR,但是,通过使用括号可以更改求值的顺序。

【例 5.17】 查询购买数量为 4 或 5 的订单信息。

```
SELECT  *
FROM  OrderInfo
WHERE  Oquantity = 4 OR Oquantity = 5;
```

3. 范围查询条件:

内含范围条件(BETWEEN...AND...):要求返回记录某个字段的值在两个指定值范围内,同时包括这两个指定的值。

排除范围条件(NOT BETWEEN...AND...):要求返回记录某个字段的值在两个指定值范围以外,且不包括这两个指定的值。

【例 5.18】 查询购买数量是 5～10(含 10)的订单信息。

方法 1:使用 BETWEEN...AND...

```
SELECT  *
FROM  OrderInfo
WHERE  Oquantity  BETWEEN 5 AND 10;
```

方法 2:使用逻辑运算符 AND。

说明:BETWEEN 5 AND 10 相当于 Oquantity≥5 AND Oquantity≤10。

【例 5.19】 查询购买数量不在 5～10 区间的订单信息。

方法 1:使用 NOT BETWEEN...AND...

```
SELECT  *
FROM  OrderInfo
WHERE  Oquantity NOT BETWEEN 5 AND 10;
```

方法 2:使用逻辑运算符 OR。

```
SELECT  *
FROM  OrderInfo
WHERE  Oquantity < 5 OR Oquantity > 10;
```

【例 5.20】 查询订单状态为已提交,且购买数量为 1～5 的订单信息。

方法 1:使用 BETWEEN...AND...

```
SELECT  *
FROM  OrderInfo
WHERE  Status = '已提交' AND Oquantity BETWEEN 1 AND 5;
```

方法 2:使用逻辑运算符 AND。

```
SELECT  *
FROM  OrderInfo
WHERE  Status = '已提交' AND Oquantity > = 1 AND Oquantity < = 5;
```

4. 列表查询条件

包含列表查询条件的查询将返回所有字段值与指定列表中某个值相匹配的记录。通常使用 IN 关键字来指定这个列表查询条件。

格式如下:

```
IN(列表值 1,列表值 2,...)
```

列表中的项目之间必须使用逗号分隔,并且括在括号中。

【例 5.21】 查询购买人手机号码为 13127 ** 3128 和 13527 ** 8060 的购买人信息,显示购买人手机号码、姓名和电子邮箱。

方法 1:使用 IN 关键字。

```
SELECT  BPhone,BName,Email
FROM  BuyerInfo
WHERE  BPhone IN('13127 ** 3128','13527 ** 8060');
```

方法 2:使用逻辑运算符 OR。

```
SELECT  BPhone,BName,Email
FROM  BuyerInfo
WHERE  BPhone = '13127 ** 3128' OR BPhone = '13527 ** 8060'
```

【例 5.22】 查询购买人手机号码不是 13127 ** 3128 和 13527 ** 8060 的购买人信息,显示购买人手机号码、姓名和电子邮箱。

```
SELECT  BPhone,BName,Email
FROM  BuyerInfo
WHERE  BPhone NOT IN('13127 ** 3128','13527 ** 8060')
```

5. 模式查找条件

模式查找条件通常通过 LIKE 和 REGEXP(或 RLIKE,它们是等价的)操作符来实现。这些操作符允许根据指定的模式来搜索数据。LIKE 操作符用于在 WHERE 子句中搜索列中的指定模式,它支持两个通配符。格式如下:

```
<字符串表达式> [NOT] LIKE <模式>
```

参数说明如下。

(1) 在 LIKE 运算符前面使用 NOT 运算符,表示对运算结果取一次反。

(2) 可用于 CHAR、VARCHAR、TEXT、NTEXT、DATETIME 和 TIMESTAMP 等类型查询。

(3) LIKE 关键字使用通配符来表示字符串需要匹配的模式。%:匹配零个或多个字符。_:匹配单个字符。

LIKE 模式如下。

%abc:匹配以 abc 结尾的任何字符串。

abc%:匹配以 abc 开头的任何字符串。

%abc%:匹配包含 abc 的任何字符串。

a_c:匹配第二个字符是任意字符且以 a 开头、c 结尾的任何三个字符的字符串。

(4) REGEXP(或 RLIKE)操作符。REGEXP 操作符提供了更强大的模式匹配功能,它基于正则表达式,允许进行更复杂的搜索,如匹配数字、字母、特定字符集等。

REGEXP 模式如下。

^[0-9]+ $:匹配仅包含数字的字符串。

[A-Z]：匹配任何大写字母。

[a-z0-9]：匹配任何小写字母或数字。

^[^0-9]：匹配不以数字开头的任何字符串。

(abc|def)：匹配 abc 或 def。

【例 5.23】 在 BuyerInfo 表中，查找姓李的购买人信息。

```
SELECT  *
FROM  BuyerInfo
WHERE  BName LIKE '李%'
```

【例 5.24】 在 BuyerInfo 表中，查找姓李、姓张和姓何的购买人信息。

```
SELECT *
FROM BuyerInfo
WHERE BName LIKE '李%' OR BName LIKE '张%' OR BName LIKE '何%';
```

或

```
SELECT *
FROM BuyerInfo
WHERE BName REGEXP '^[李张何]';
```

注意：REGEXP 是一个运算符，用于进行正则表达式匹配。^是一个锚字符，表示匹配行的开始。[李张何] 是一个字符类，表示匹配方括号内的任意一个字符。

【例 5.25】 在 BuyerInfo 表中，查询姓名中含有“秀”字的购买人信息。

```
SELECT  *
FROM  BuyerInfo
WHERE  BName LIKE '%秀%';
```

【例 5.26】 在 BuyerInfo 表中，查询姓名第二个字为“阳”字的购买人信息。

```
SELECT  *
FROM  BuyerInfo
WHERE  BName LIKE '_阳%';
```

【例 5.27】 在 BuyerInfo 表中，查询购买人手机号码末位数字不在 5～9 范围内的购买人信息。

```
SELECT *
FROM BuyerInfo
WHERE BPhone REGEXP '[^5-9]$';
```

注意：这里的正则表达式[^5-9]$ 意味着匹配任何不以 5、6、7、8 或 9 结尾的字符串。$ 是一个锚点，表示字符串的末尾。

6. 空值判断查询条件

空值判断查询条件常用来查询某一字段值为空值的记录，可以使用 IS NULL 或 IS NOT NULL 关键字来指定这种查询条件。

注意：NULL 值表示字段的数据值未知或不可用，它并不表示零(数字值或二进制值)、零长度的字符串或空白(字符值)。

【例 5.28】 查询农户基本信息为空值的农户信息记录。

```
SELECT   *
FROM  FarmerInfo
WHERE  F_BasicInfo IS NULL;
```

【例 5.29】 查询农户基本信息不为空值的农户信息记录。

```
SELECT   *
FROM  FarmerInfo
WHERE  F_BasicInfo IS NOT NULL;
```

7. 对查询结果进行排序

在 SELECT 语句中，可以使用 ORDER BY 子句对查询返回的结果进行排序。

语法格式如下：

```
ORDER BY COLUMN_NAME [ASC|DESC] [,...n]
```

参数说明如下。

COLUMN_NAME：需排序的列名。

ASC 表示升序排序，为默认值；DESC 为降序排序。

n：可以选择多列进行排序。

【例 5.30】 查询入库数量为 30 的农产品采购信息，并按购入时间先后排序。

```
SELECT   *
FROM  ASupplyInfo
WHERE Aquantity = 30
ORDER BY Atime ASC;
```

【例 5.31】 查询类别为水果的农产品信息，并按售价由小到大排序，售价相同按库存量降序排序。

```
SELECT *
FROM  AProductInfo
WHERE  Category = '水果'
ORDER BY  OutPrice ASC, Inventory DESC;
```

【例 5.32】 在 AProductInfo 表中，查询类别为水果的库存量最高的前两名。

```
SELECT *
FROM  AProductInfo
WHERE  Category = '水果'
ORDER BY Inventory DESC
LIMIT 2;
```

5.3 统计农产品的销售信息

通过数据分析，用户可以从大量信息中总结规律、识别趋势，并生成具有参考价值的数据报表。对于农产品经营者来说，通过分析成本、利润、最高与最低销售量等关键指标，能够有效支持业务决策。

为协助用户对数据库中的记录进行统计汇总，MySQL 提供了一系列聚合函数。本节将系统介绍这些函数的作用与使用方法，具体包括对数据进行汇总、计算某一字段的平均

值、统计符合特定条件的记录数量，以及查询某个字段中的最大值和最小值。

5.3.1 GROUP BY 子句

按指定的条件进行分类汇总，并且如果 SELECT 子句中包含聚合函数，则计算每组的汇总值。

语法格式如下：

```
GROUP BY GROUP_BY_EXPRESSION [,...n]
```

参数说明如下。

GROUP_BY_EXPRESSION：是对其执行分组的表达式，也称为分组列。在选择列表(SELECT 子句)中定义的列的别名不能用于指定分组列。

在使用 GROUP BY 子句时，SELECT 子句中通常只能包含聚合函数、GROUP BY 中指定的列，或者是这些列的常量表达式。

1. 聚合函数

在数据库的数据查询分析中，聚合函数扮演着举足轻重的角色。它们的主要功能是在查询结果集中生成汇总值。除了 COUNT(*)之外，其他聚合函数会处理选定列中的所有值，并最终生成一个汇总结果。这些函数既可以作用于整张表的所有行，也可以针对 WHERE 子句所指定的子集，或者表中按照特定条件分组的一组或多组行。每当应用聚合函数时，它都会为每组行计算出一个值。

1) 求和函数 SUM()

语法格式如下：

```
SUM([ALL | DISTINCT] EXPRESSION)
```

参数说明如下。

ALL：表示对所有选定值进行求和，为默认值，通常可以省略。

DISTINCT：表示仅对唯一值进行求和，即排除重复值后的求和。

EXPRESSION：代表要求和的数字表达式，它可以是列名、数学表达式或计算字段等。

结果：返回数字表达式中所有值的和。

查询语句的语法格式如下：

```
SELECT SUM(COLUMN_NAME)
FROM TABLE_NAME
[WHERE CONDITION]
[GROUP BY GROUP_BY_EXPRESSION];
```

参数说明如下。

COLUMN_NAME：需要求和的数值型列名。

TABLE_NAME：包含目标列的表名。

CONDITION(可选)：用于指定哪些行应包含在求和操作中。

GROUP BY GROUP_BY_EXPRESSION(可选)：用于根据一个或多个表达式对行进行分组，并为每个组分别计算总和。

【例 5.33】 从 AProductInfo 表中统计水果的库存量。查询结果如图 5.4 所示，代码如下：

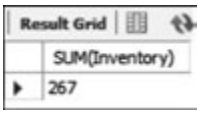


图 5.4 统计水果的库存量的查询结果

```
SELECT  SUM(Inventory)
FROM  AProductInfo
WHERE  Category = '水果';
```

说明：在 MySQL 中，当使用聚合函数（如 SUM()）而不为结果列指定别名时，MySQL 通常会为结果集中的列分配一个默认列名。然而，这个默认列名可能不是很有描述性，它通常会是聚合函数本身的一个变种显示，但这取决于具体的 MySQL 版本和配置。

2）求平均函数 AVG()

语法格式如下：

```
AVG([ALL | DISTINCT] EXPRESSION)
```

ALL（默认）：计算表达式中所有值的平均值。

DISTINCT：仅计算表达式中不同（唯一）值的平均值。

EXPRESSION：可以是列名或计算表达式，结果必须是数值类型。

结果：返回指定表达式的所有（或唯一）值的平均值。

3）计数函数 COUNT()

语法格式如下：

```
COUNT([ALL | DISTINCT] EXPRESSION)
```

ALL（默认）：计算表达式中非空值的个数。

DISTINCT：仅计算表达式中不同（唯一）非空值的个数。

EXPRESSION：可以是列名或表达式。

结果：返回表达式中非空值的个数（对于 ALL）或不同非空值的个数（对于 DISTINCT）。

语法格式如下：

```
COUNT ( * )
```

*：计算选定行的总数，包括所有列。

结果：返回表中的行数（不包括任何 NULL 值行，因为 COUNT(*) 计算的是行的存在性，而不是具体的列值）。

【例 5.34】 从 AProductInfo 表中统计水果平均售价，并统计水果的品种数。查询结果如图 5.5 所示，代码如下：



图 5.5 统计水果平均售价的查询结果

```
SELECT  AVG(OutPrice) 水果平均售价,COUNT(Category) 水果品种数
FROM  AProductInfo
WHERE  Category = '水果';
```

4）求最大值函数 MAX()

语法格式如下：

```
MAX(EXPRESSION)
```

结果：返回表达式中的最大值。

5）求最小值函数 MIN()

语法格式如下：

```
MIN(EXPRESSION)
```

结果：返回表达式中的最小值。

【例 5.35】 在 AProductInfo 表中，查询水果的最高进货价和最低进货价。查询结果如图 5.6 所示，代码如下：

```
SELECT  MAX(InPrice) 最高进货价,MIN(InPrice) 最低进货价
FROM  AProductInfo
WHERE  Category = '水果';
```



图 5.6　查询水果的最高进货价和最低进货价的查询结果

聚合函数既可以单独查询，也可以在一条查询语句中同时统计。

【例 5.36】 从 AProductInfo 表中，统计蔬菜的总库存量、平均售价、蔬菜品种数、最高进货价和最低进货价。查询结果如图 5.7 所示，代码如下：

```
SELECT
   SUM(Inventory) AS 蔬菜总库存量,
   CAST(AVG(OutPrice) AS DECIMAL(4,1)) AS 平均售价,
   COUNT(DISTINCT Aname) AS 蔬菜品种数,
   MAX(InPrice) AS 最高进货价,
   MIN(InPrice) AS 最低进货价
FROM AProductInfo
WHERE Category = '蔬菜';
```

蔬菜总库存量	平均售价	蔬菜品种数	最高进货价	最低进货价
233	4.5	4	3.00	1.50

图 5.7　综合查询结果

2. GROUP BY 和聚合函数

聚合函数（如 SUM()、AVG()、COUNT()、MAX()、MIN()等）用于对一组值执行计算，并返回单个值。当与 GROUP BY 子句一起使用时，这些函数会对每个分组中的值进行计算。GROUP BY 有一个原则，就是 SELECT 后面的所有列中，没有使用聚合函数的列，必须出现在 GROUP BY 后面。

【例 5.37】 统计各类别农产品的平均进货价。查询结果如图 5.8 所示。代码如下：

```
SELECT Category AS 类别, CAST(AVG(InPrice) AS DECIMAL(10,1)) AS 平均进货价
FROM AProductInfo
GROUP BY Category;
```

【例 5.38】 统计各类别农产品的总库存量、各类别的品种数和平均售价。查询结果如图 5.9 所示。代码如下：

```
SELECT Category AS 类别, SUM(Inventory) AS 总库存量, COUNT( * ) AS 各类别的品种数,
                CAST(AVG(OutPrice) AS DECIMAL(10,1)) AS 平均售价 -- 根据需要调整精度
FROM  AProductInfo
GROUP BY Category;
```

类别	平均进货价
蔬菜	2.5
水果	8.0
禽蛋	29.5

图 5.8　按类别分类计算平均进货价的查询结果

类别	总库存量	各类别的品种数	平均售价
水果	267	3	11.7
禽蛋	221	4	39.0
蔬菜	233	4	4.5

图 5.9　按类别分类汇总

3. HAVING 子句

HAVING 子句指定组或聚合的搜索条件，并与 GROUP BY 子句一起使用。不使用 GROUP BY 时，HAVING 子句实际上没有意义，因为此时没有组可供筛选，而非像 WHERE 子句那样工作。WHERE 子句在分组前过滤数据，且不能包含聚合函数。HAVING 子句则在分组后过滤数据，并允许使用聚合函数。此外，HAVING 子句不仅限于引用 SELECT 语句中已定义的列，还可以直接包含聚合函数表达式，但无须“重复”这些表达式，除非它们未在 SELECT 列表中作为别名明确列出。

【例 5.39】 查询各类别农产品平均进货价高于 5 的农产品记录，使用 HAVING 子句。查询结果如图 5.10 所示。代码如下：

```
SELECT Category AS 类别, CAST(AVG(InPrice) AS DECIMAL(10,1)) AS 平均进货价,
                COUNT( * ) AS 农产品数量
FROM AProductInfo
GROUP BY Category
HAVING AVG(InPrice) > 5;
```

【例 5.40】 查询不同类别的农产品中，进货价高于 5 的农产品数量达到 2 种以上的记录。查询结果如图 5.11 所示。代码如下：

```
SELECT Category AS 类别, COUNT( * ) AS 进货价高于 5 的农产品数量
FROM  AProductInfo
WHERE InPrice > 5
GROUP BY Category
HAVING COUNT( * ) > 2;
```

类别	平均进货价	农产品数量
水果	8.0	3
禽蛋	29.5	4

图 5.10　HAVING 子句的查询结果

类别	进货价高于5的农产品数量
水果	3
禽蛋	4

图 5.11　HAVING 子句查询农产品进货价高于 5 的查询结果

5.3.2　使用 UNION 子句

UNION 是数据库 SQL 中的运算符，可以将两个或多个 SELECT 语句的结果合成一个结果集。这与使用连接组合两个表中的列不同。使用 UNION 组合两个查询的结果集的两个基本规则如下。

（1）使用 UNION 组合的结果集都必须具有相同的结构，且列数相同。

（2）相应的结果集列的数据类型必须兼容。

注意：①UNION 中的每个查询所涉及数据的列必须具有相同的列数、相同的数据类型，并以相同的顺序出现。②最后结果集中的列名来自第一个 SELECT 语句。③若 UNION 中包含 ORDER BY 子句，则将对最后的结果集排序。④在合并结果集中时，默认从最后的结果集中删除重复的行，除非使用 ALL 关键字。

UNION 运算符的指定格式如下：

```
SELECT column1, column2 FROM table1
UNION [ALL]
SELECT column1, column2 FROM table2;
```

【例 5.41】 使用 UNION 统计 2024 年 3 月 5 日各农产品的购买数量，购买数量小于 3 为产品滞销，[4,7]为产品销量一般，[8,∞)为产品畅销。查询结果如图 5.12 所示。代码如下：

```
SELECT '产品滞销' AS 等级, '[1,3]' AS 数量区间, COUNT( * ) AS 销量情况
FROM OrderInfo
WHERE DATE(SubTime) = '2024 - 03 - 05' AND Oquantity BETWEEN 1 AND 3
UNION
SELECT '产品销量一般' AS 等级, '[4,7]' AS 数量区间, COUNT( * ) AS 销量情况
FROM OrderInfo
WHERE DATE(SubTime) = '2024 - 03 - 05' AND Oquantity BETWEEN 4 AND 7
UNION
SELECT '产品畅销' AS 等级, '[8,∞)' AS 数量区间, COUNT( * ) AS 销量情况
FROM OrderInfo
WHERE DATE(SubTime) = '2024 - 03 - 05' AND Oquantity > 8;
```



图 5.12　UNION 的查询结果

5.4　高级查询

5.4.1　关联表查询

一个数据库中的多个表之间一般都存在某种内在联系，它们共同提供有用的信息。前文的简单查询和统计都是针对一个表进行的。例如，订单信息表中没有建立购买人姓名字段，所以无法在订单信息查询中提供购买人姓名数据。数据库中的各个表存放着不同的数据，往往需要用多个表中的数据来组合查询出所需要的信息。所谓多表查询是相对单表而言的，指从多个数据表中查询数据。

若一个查询同时涉及两个以上的表，称之为连接查询。连接查询是关系数据库中最主要的查询。表之间的连接是通过相等的字段值连接起来的查询称为等值连接查询。等值多表查询将按照等值的条件查询多个数据表中关联的数据。要求关联的多个数据表的某些字段具有相同的属性，即具有相同的数据类型、宽度和取值范围。这里介绍如何使用

WHERE 子句关联表实现多表查询。

【例 5.42】 查询农产品采购信息和农户信息。查询结果如图 5.13 所示。代码如下：

```
SELECT *
FROM  ASupplyInfo,FarmerInfo
WHERE  ASupplyInfo.FPhone = FarmerInfo.FPhone;
```

AID	FPhone	Atime	Aquantity	FPhone	FName	F_BasicInfo
1	133275**001	2024-01-24 09:00:00	80	13327**0001	张三丰	主要供应苹果、雪梨等水果
2	138275**099	2024-01-25 08:45:00	30	13827**0099	李天一	主要供应白牛奶葡萄
3	139275**088	2024-01-26 14:20:00	50	13927**0088	王天佑	主要供应杏花鸡
4	139275**068	2024-02-01 11:15:00	75	13927**6068	陈彩云	主要供应青菜
5	133275**009	2024-02-01 13:30:00	50	13327**7009	梁浩朝	主要供应土豆
6	139275**088	2024-02-01 15:45:00	30	13927**0088	王天佑	主要供应杏花鸡
7	133275**001	2024-01-24 10:30:00	100	13327**0001	张三丰	主要供应苹果、雪梨等水果

图 5.13　关联表的查询结果

从图 5.13 的查询结果可以看出，显示结果是两张表的所有记录，其中农户电话列重复出现，在查询时，可以根据实际情况选择显示相关字段。

【例 5.43】 查询采购农户张三丰的信息，显示字段包含农户电话、农户姓名、农产品名称和入库数量。查询结果如图 5.14 所示。代码如下：

方法 1：

```
SELECT  FarmerInfo.FPhone,FName,AName,Aquantity
FROM  ASupplyInfo,FarmerInfo
WHERE ASupplyInfo.FPhone = FarmerInfo.FPhone AND FName = '张三丰';
```

Result Grid　Filter Rows:

FPhone	FName	AName	Aquantity
13327510001	张三丰	红富士	80
13327510001	张三丰	蛇果	100

图 5.14　带条件的关联表查询结果

方法 2：使用表别名。

```
SELECT  F.FPhone,FName,AName,Aquantity
FROM  ASupplyInfo CS,FarmerInfo F
WHERE  CS.FPhone = F.FPhone AND FName = '张三丰';
```

在实际应用中，需要将多个数据表关联查询，超过两个数据表的关联查询称为多表查询。

【例 5.44】 查询购买人张飞购买农产品的信息，显示字段包含购买人手机号码、购买人姓名、农产品名称、售价和购买数量。查询结果如图 5.15 所示。代码如下：

```
SELECT
   b.BPhone AS 购买人手机号码,
   b.BName AS 购买人姓名,
   p.AName AS 农产品名称,
   p.OutPrice AS 售价,
   o.Oquantity AS 购买数量
FROM
   BuyerInfo b
INNER JOIN
   OrderInfo o ON b.BID = o.BID
```

```
INNER JOIN
   AProductInfo p ON o.AID = p.AID
WHERE
   b.Bname = '张飞';
```

购买人手机号码	购买人姓名	农产品名称	售价	购买数量
13527**8060	张飞	红富士	12.00	5

图 5.15　多表查询结果

在单表采购信息表的统计中,可以统计农户销售农产品的总数量。但统计结果没有包含农户姓名字段,因为采购信息表不包含农户姓名字段,农户姓名字段属于农户信息表,所以必须先将采购信息表和农户信息表通过农户电话字段关系查询,然后进行关联表统计即可。

【例 5.45】 统计农户销售农产品的总数量,结果包含农户电话、农户姓名和农户售出农产品数量字段信息。查询结果如图 5.16 所示。代码如下:

```
SELECT   f.FPhone AS 农户电话, f.FName AS 农户姓名, SUM(s.Aquantity) AS 农户售出农产品数量
FROM   ASupplyInfo s,FarmerInfo  f
WHERE   s.FPhone = f.FPhone
GROUP BY   f.FPhone, f.FName;
```

【例 5.46】 统计购买人购买农产品消费了多少钱。结果包含购买人姓名、农产品名称和费用字段信息,查询结果如图 5.17 所示。代码如下:

```
SELECT     BuyerInfo.BName, AProductInfo.AName,
           SUM(AProductInfo.OutPrice * OrderInfo.Oquantity) AS 费用
FROM   BuyerInfo, OrderInfo, AProductInfo
WHERE     BuyerInfo.BID = OrderInfo.BID     AND OrderInfo.AID = AProductInfo.AID
GROUP BY   BuyerInfo.BName, AProductInfo.AName;
```

农户电话	农户姓名	农户售出农产品数量
13327**0001	张三丰	180
13827**0099	李天一	30
13927**0088	王天佑	80
13927**6068	陈彩云	75
13327**7009	梁浩朝	50

图 5.16　分组的关联表查询结果

BName	AName	费用
张飞	红富士	60
李明	白牛奶葡萄	60
梁秀丽	杏花鸡	104
何小莉	宁夏菜心	10
李剑雄	腊牛肉	120
欧阳松松	西兰花	60

图 5.17　统计的关联表的查询结果

5.4.2　子查询

1. 子查询的概念

在 MySQL 中,一个 SELECT...FROM...WHERE 语句称为一个查询块。将一个查询块嵌套在另一个查询的查询条件(WHERE 或 HAVING 子句)之中时称为嵌套查询。嵌套查询是指在一个外层查询中包含另一个内层查询,其中,外层查询称为主查询,内层查询称为子查询。通常情况下,使用嵌套查询中的子查询先挑选出部分数据,作为主查询的数据来源或搜索条件。子查询总是写在圆括号中,任何允许使用表达式的地方都可以使用子查询。

子查询的求解方法是由里向外处理。即每个子查询在其上一级查询处理之前求解,子查询的结果用于建立其主查询的查找条件。

子查询可以用一系列简单查询构成复杂的查询，从而明显增强了 SQL 的查询能力。

下面是有关子查询的几点说明。

(1) 子查询通常需要包括以下组件。

- 包含的标准选择列表组件的标准 SELECT 查询。
- 包含一个或多个表或者视图名的标准 FROM 子句。
- 可选的 WHERE 子句。
- 可选的 GROUP BY 子句。
- 可选的 HAVING 子句。

(2) 子查询的 SELECT 语句通常使用圆括号括起来。

(3) 子查询的 SELECT 语句中不能包含 COMPUTE 子句。

(4) 除非在子查询中使用了 LIMIT 子句，否则子查询中不能包含 ORDER BY 子句。

(5) 子查询可以嵌套在外部的 SQL 语句的 WHERE、HAVING、SELECT 子句内，以及 IN、EXIST 等条件中。

(6) 如果某个数据表只出现在子查询中，而不出现在主查询中，那么在数据列表中不能包含数据表中的字段。

(7) 包含子查询的语句通常采用以下格式：

```
WHERE 表达式 [NOT] IN (子查询)
WHERE 表达式  比较运算符[ANY | ALL](子查询)
WHERE [NOT] EXISTS (子查询)
```

基本子查询的语法格式如下：

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name [comparison_operator]
(SELECT subquery_column FROM subquery_table WHERE subquery_condition);
```

参数说明如下。

- column1，column2，...：要从主查询中选择的列。
- table_name：主查询要查询的表。
- column_name：主查询中用于与子查询结果进行比较的列。
- comparison_operator：比较运算符，如＝、＞、＜、IN、EXIST 等。
- (SELECT subquery_column FROM subquery_table WHERE subquery_condition)：子查询，返回一个或多个值或结果集，用于与主查询中的列进行比较。

2. 子查询的应用

1) 带有 IN 关键字的子查询

带有 IN 关键字的子查询是指主查询与子查询之间用 IN 进行连接，即判断某个属性列值是否存在子查询的结果中。由于子查询的结果往往是一个集合，所以 IN 是嵌套查询中最常用的关键字。

带有 IN 关键字的子查询的语法格式如下：

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name IN (SELECT subquery_column FROM subquery_table WHERE subquery_condition);
```


这个格式用于检查 column_name 的值是否存在于子查询返回的结果集中。

【例 5.47】 查询采购农产品的入库数量高于 30 的农产品信息。查询结果如图 5.18 所示。代码如下：

```
SELECT *
FROM AProductInfo
WHERE AID IN (
   SELECT AID
   FROM ASupplyInfo
   WHERE Aquantity > 30
);
```

AID	AName	OutPrice	InPrice	Category	Inventory
1	红富士	12.00	8.00	水果	100
3	杏花鸡	26.00	20.00	禽畜	90
4	宁夏菜心	5.00	3.00	蔬菜	68
5	土豆	3.00	1.50	蔬菜	75
7	蛇果	13.00	9.00	水果	87

图 5.18 带有 IN 关键字的子查询结果

【例 5.48】 查询购买农产品的购买人信息，显示购买人手机号码、购买人姓名和电子邮箱。

```
SELECT BPhone, BName, Email
FROM BuyerInfo
WHERE BID IN (
   SELECT BID FROM OrderInfo
);
```

除了 IN 关键字，还可以使用 NOT IN 关键字来进行列表查询。NOT IN 的含义与 IN 正好相反，查询结果将返回不在列表范围内的所有记录。

2）带有比较运算符的子查询

带有比较运算符的子查询指主查询与子查询之间用比较运算符进行连接。当用户知道内层查询返回的是单值时，可以用＞、＞＝、＜、＜＝、＝、＜＞等比较运算符。返回单值子查询，只返回一行一列。

【例 5.49】 查询农产品进货价高于农产品售价的农产品信息。代码如下：

```
SELECT
    AID, AName AS 农产品名称, InPrice AS 进货价, OutPrice AS 售价,
   Category AS 类别, Inventory AS 库存量
FROM AProductInfo
WHERE InPrice > OutPrice;
```

3）带有 ANY 或 ALL 关键字的子查询

带有 ANY 和 ALL 关键字的子查询用于将主查询中的列与子查询返回的结果集中的每个值进行比较。ANY 表示与结果集中的任何一个值匹配即可，而 ALL 表示必须与结果集中的所有值都匹配。

带有 ANY 或 ALL 关键字的子查询的语法格式为：

```
-- 使用 ANY 关键字
SELECT column1, column2, ...
FROM table_name
WHERE column_name comparison_operator ANY (SELECT subquery_column FROM subquery_table WHERE
subquery_condition);
-- 使用 ALL 关键字
SELECT column1, column2, ...
FROM table_name
WHERE column_name comparison_operator ALL (SELECT subquery_column FROM subquery_table WHERE
subquery_condition);
```

【例 5.50】 查询农产品售价低于任一类别的农产品的平均售价的农产品信息。查询结果如图 5.19 所示。代码如下：

方法 1：

```
SELECT * FROM  AProductInfo
WHERE  OutPrice < ANY(
SELECT AVG(OutPrice) FROM AProductInfo
GROUP BY Category)
```

AID	AName	OutPrice	InPrice	Category	Inventory
1	红富士	12.00	8.00	水果	100
2	白牛奶葡萄	10.00	7.00	水果	80
3	杏花鸡	26.00	20.00	禽畜	90
4	宁夏菜心	5.00	3.00	蔬菜	68
5	土豆	3.00	1.50	蔬菜	75
7	蛇果	13.00	9.00	水果	87
8	黑土猪	30.00	20.00	禽畜	30
9	西兰花	6.00	3.00	蔬菜	40
10	豆角	4.00	2.50	蔬菜	50

图 5.19　例 5.50 的查询结果

ANY 与 SOME 等价。

方法 2：

```
SELECT *   FROM  AProductInfo
WHERE  OutPrice < some(
SELECT AVG(OutPrice) FROM AProductInfo
GROUP BY Category)
```

【例 5.51】 查询农产品售价低于所有类别的农产品的平均售价的农产品信息。查询结果如图 5.20 所示。代码如下：

```
SELECT  * FROM  AProductInfo
WHERE  OutPrice < ALL(
SELECT AVG(OutPrice) FROM AProductInfo
GROUP BY Category)
```

AID	AName	OutPrice	InPrice	Category	Inventory
5	土豆	3.00	1.50	蔬菜	75
10	豆角	4.00	2.50	蔬菜	50

图 5.20　例 5.51 的查询结果

【例 5.52】 查询所有农产品中售价最高的农产品信息。代码如下：

```
SELECT * FROM AProductInfo
WHERE OutPrice >= ALL(SELECT OutPrice FROM AProductInfo)
```

使用单值比较运算符，执行结果与 ALL 相同。

```
SELECT * FROM AProductInfo
WHERE OutPrice >= (SELECT MAX(OutPrice) FROM AProductInfo)
```

4）带有 EXISTS 关键字的子查询

带有 EXISTS 关键字的子查询不返回任何数据，只产生逻辑真假值，即若子查询非空，则返回 TRUE，从而主查询的 WHERE 子句返回 TRUE。EXISTS 子查询用于检查子查询是否返回至少一行数据。如果返回至少一行，则 EXISTS 条件为真。

语法格式如下：

```
SELECT column1, column2, ...
FROM table_name t1
WHERE EXISTS (SELECT 1 FROM subquery_table t2 WHERE t1.common_column = t2.common_column AND
subquery_condition);
```

【例 5.53】 查询 OrderInfo 表中购买数量高于 5 的农产品信息。查询结果如图 5.21 所示。代码如下：

```
SELECT * FROM AProductInfo
WHERE EXISTS (
   SELECT * FROM OrderInfo
   WHERE AProductInfo.AID = OrderInfo.AID AND Oquantity > 5
);
```

AID	AName	OutPrice	InPrice	Category	Inventory
2	白牛奶葡萄	10.00	7.00	水果	80
9	西兰花	6.00	3.00	蔬菜	40

图 5.21 例 5.53 的查询结果

5.4.3 连接查询

如果连接查询确实涉及两个或更多表，通过它们之间的共同列（通常是主键和外键）来关联数据，就称为连接查询。连接查询的结果集或结果表，称为表之间的连接。连接查询实际上是通过各个表之间共同列的关联性来查询数据的，它是关系数据库中最主要的查询。

连接查询分为等值连接查询、非等值连接查询、自连接查询、外部连接查询和复合连接查询。

SQL-92 标准所定义的 FROM 子句的连接语法格式为：

```
FROM  JOIN_TABLE  JOIN_TYPE  JOIN_TABLE
[ON (JOIN_CONDITION)]
```

参数说明如下。

JOIN_TABLE：指出参与连接操作的表名，连接可以对同一个表操作，也可以对多表操作，对同一个表操作的连接又称自连接。

JOIN_TYPE：指出连接类型，可分为三种：内连接、外连接和交叉连接。

ON (JOIN_CONDITION)：连接操作中的 ON(JOIN_CONDITION)子句指定连接条件，它是由列名、比较运算符和逻辑运算符组成的表达式，用于确定哪些行应该被连接起来。

1. 内连接查询

内连接查询(INNER JOIN)是通过比较两个或多个表之间的某些列的数据来查询满足连接条件的数据行的操作。在内连接中，只有满足连接条件的元组(即行)才会出现在结果集中。根据比较方式的不同，内连接可以分为等值连接、自然连接和非等值连接。

1) 等值连接(Equijoin)

等值连接指在连接条件中使用等于(=)运算符来比较被连接列的列值。在查询结果中，会列出被连接表中的所有列，包括可能的重复列(如果两个表中有同名的列，这些列都会在结果集中出现，且各自保留其原始值)。值得注意的是，虽然等值连接是内连接的一种常见形式，但内连接本身并不局限于等值比较。

2) 非等值连接(Non-equijoin)

非等值连接指在连接条件中使用除等于运算符以外的其他比较运算符来比较被连接列的列值。这些运算符包括但不限于大于(>)、大于或等于(>=)、小于(<)、小于或等于(<=)、不等于(<> 或 !=)。非等值连接允许更复杂的连接条件，从而能够查询出更灵活的数据关系。

3) 自然连接(Natural Join)

自然连接是一种特殊的等值连接，它要求连接的两个关系(即表)具有公共属性(即同名的列)，并且在连接条件中自动使用这些公共属性进行等值比较。与自然连接不同，普通的等值连接需要显式地指定连接条件。在自然连接的结果集中，相同的公共属性只会出现一次，即这些列会被合并为一个列，并且只包含满足连接条件的唯一值。然而，需要注意的是，在某些数据库系统中，自然连接可能会忽略 NULL 值，这可能导致意外的结果。因此，在使用自然连接时需要谨慎考虑数据的完整性和准确性。

内连接查询的语法如下：

```
SELECT SELECT_LIST
FROM TABLE1
INNER JOIN TABLE2
ON TABLE1.COLUMN_NAME = TABLE2.COLUMN_NAME
[INNER JOIN TABLE3
ON TABLE2.COLUMN_NAME = TABLE3.COLUMN_NAME
...]
[WHERE SEARCH_CONDITION]
[ORDER BY ORDER_LIST] [LIMIT ROW_COUNT];
```

参数说明如下。

- SELECT SELECT_LIST：指定要查询的列。SELECT_LIST 可以包含表中的一个或多个列，以及可能的聚合函数或表达式。
- FROM TABLE1：指定查询的主表(即第一个表)。
- INNER JOIN TABLE2 ON TABLE1.COLUMN_NAME=TABLE2.COLUMN_NAME：指定要与主表进行内连接的第二个表及连接条件。连接条件通常是两个

表中某列之间的等值比较。

- [INNER JOIN TABLE3 ON TABLE2.COLUMN_NAME = TABLE3.COLUMN_NAME ...]：这是一个可选部分，表示可以进一步与其他表进行内连接，每个连接都需要指定连接条件。
- [WHERE SEARCH_CONDITION]：这是一个可选的 WHERE 子句，用于指定额外的筛选条件，以进一步限制查询结果。
- [ORDER BY ORDER_LIST]：这是一个可选的 ORDER BY 子句，用于指定结果集的排序方式。
- [LIMIT ROW_COUNT]：这是一个可选的 LIMIT 子句，用于限制返回的行数。

【例 5.54】 查询农产品信息和订单信息。查询结果如图 5.22 所示。代码如下：

```
SELECT
   A.AName, A.OutPrice, A.InPrice,A.Category,A.Inventory,
   O.Number,  O.SubTime, O.Status, O.BID,O.Oquantity
FROM  AProductInfo A INNER JOIN  OrderInfo O
 ON A.AID = O.AID;
```

AName	OutPrice	InPrice	Category	Inventory	Number	SubTime	Status	BID	Oquantity
红富士	12.00	8.00	水果	100	20240201000000001	2024-03-05 08:11:05	已完成	B1	5
白牛奶葡萄	10.00	7.00	水果	80	20240201000000002	2024-03-05 08:18:25	已完成	B2	6
杏花鸡	26.00	20.00	禽畜	90	20240201000000003	2024-03-06 10:10:20	已发货	B3	4
宁夏菜心	5.00	3.00	蔬菜	68	20240201000000004	2024-03-07 11:12:21	已提交	B4	2
鲜牛肉	60.00	50.00	禽畜	46	20240201000000005	2024-03-07 11:18:30	已提交	B5	2
西兰花	6.00	3.00	蔬菜	40	20240201000000006	2024-03-08 10:10:21	已提交	B6	10

图 5.22　内连接的查询结果

【例 5.55】 查询红富士的订单信息，显示字段包含农产品名称、售价、进货价、订单状态和购买数量。代码如下：

```
SELECT   A.AName,A.OutPrice,A.InPrice, O.Status, O.Oquantity
FROM   AProductInfo A  INNER JOIN   OrderInfo O ON A.AID = O.AID
WHERE    A.AName = '红富士';
```

2. 外连接查询

外连接分为左外连接（LEFT OUTER JOIN 或 LEFT JOIN）、右外连接（RIGHT OUTER JOIN 或 RIGHT JOIN）。与内连接不同，外连接不仅列出满足连接条件的行，还会列出左表（在左外连接时）和右表（在右外连接时）。对于不满足连接条件的行，结果集中相应的列将包含空值（NULL）。

语法格式如下：

```
SELECT SELECT_LIST
FROM TABLE1
LEFT [OUTER] JOIN TABLE2
ON TABLE1.COLUMN_NAME = TABLE2.COLUMN_NAME
[WHERE SEARCH_CONDITION]
[ORDER BY ORDER_LIST]
[LIMIT ROW_COUNT];
```

参数说明如下。

- SELECT_LIST：指定要查询的列。

- TABLE1、TABLE2：要连接的表。
- LEFT［OUTER］JOIN TABLE2：指定左外连接。OUTER 关键字是可选的。
- ON TABLE1.COLUMN_NAME＝TABLE2.COLUMN_NAME：指定连接条件。
- ［WHERE SEARCH_CONDITION］：是可选的，用于指定额外的筛选条件。
- ［ORDER BY ORDER_LIST］：是可选的，用于指定结果集的排序方式。
- ［LIMIT ROW_COUNT］：是可选的，用于限制返回的行数。

1）左外连接

左外连接的结果集包括左表(TABLE1)的所有行。如果左表的某行在右表(TABLE2)中没有匹配的行，则结果集中该行的右表列将包含空值(NULL)。这是左外连接与内连接的主要区别之一。

【例 5.56】 使用左外连接查询所有农户信息和采购信息。查询结果如图 5.23 所示。代码如下：

```
SELECT *
FROM  FarmerInfo F LEFT OUTER JOIN ASupplyInfo B
ON F.FPhone = B.FPhone
```

FPhone	FName	F_BasicInfo	AID	FPhone	Atime	Aquantity
13327**0001	张三丰	主要供应苹果、雪梨等水果	1	13327**0001	2024-01-24 09:00:00	80
13327**0001	张三丰	主要供应苹果、雪梨等水果	7	13327**0001	2024-01-24 10:30:00	100
13327**0056	王磊	主要供应西兰花	NULL	NULL	NULL	NULL
13327**3679	陈添	主要供应山羊肉	NULL	NULL	NULL	NULL
13327**7009	梁浩朝	主要供应土豆	5	13327**7009	2024-02-01 13:30:00	50
13427**7010	梁向阳	主要供应豆角	NULL	NULL	NULL	NULL
13827**3211	张天明	主要供应牛肉	NULL	NULL	NULL	NULL
13827**0099	李天一	主要供应白牛奶葡萄	2	13827**0099	2024-01-25 08:45:00	30
13927**0088	王天佑	主要供应杏花鸡	3	13927**0088	2024-01-26 14:20:00	50
13927**0088	王天佑	主要供应杏花鸡	6	13927**0088	2024-02-01 15:45:00	30
13927**6068	陈彩云	主要供应青菜	4	13927**6068	2024-02-01 11:15:00	75
15007**7001	李天一	主要供应巨峰葡萄	NULL	NULL	NULL	NULL

图 5.23　左外连接的查询结果

2）右外连接

右外连接是左外连接的反向连接。它返回右表中的所有行，以及左表中满足连接条件的行。如果右表的某行在左表中没有匹配的行，则结果集中该行的左表列将包含空值(NULL)。

【例 5.57】 使用右外连接查询所有农户信息和采购信息。查询结果如图 5.24 所示。代码如下：

```
SELECT  *
FROM  FarmerInfo F RIGHT OUTER JOIN ASupplyInfo B
ON  F.FPhone = B.FPhone
```

FPhone	FName	F_BasicInfo	AID	FPhone	Atime	Aquantity
13327**0001	张三丰	主要供应苹果、雪梨等水果	1	13327**0001	2024-01-24 09:00:00	80
13827**0099	李天一	主要供应白牛奶葡萄	2	13827**0099	2024-01-25 08:45:00	30
13927**0088	王天佑	主要供应杏花鸡	3	13927**0088	2024-01-26 14:20:00	50
13927**6068	陈彩云	主要供应青菜	4	13927**6068	2024-02-01 11:15:00	75
13327**7009	梁浩朝	主要供应土豆	5	13327**7009	2024-02-01 13:30:00	50
13927**0088	王天佑	主要供应杏花鸡	6	13927**0088	2024-02-01 15:45:00	30
13327**0001	张三丰	主要供应苹果、雪梨等水果	7	13327**0001	2024-01-24 10:30:00	100

图 5.24　右外连接的查询结果

3. 交叉连接查询

交叉连接(CROSS JOIN)是一种特殊的连接类型,它返回两个表中所有数据行的笛卡儿积。这意味着两个表中的每一行都会与另一个表中的每一行进行组合,生成一个包含所有可能组合的结果集。结果集合中的数据行数等于第一个表中的行数乘以第二个表中的行数。

1) 关系名限定同名属性

如果两个表中有同名的属性,并且需要在查询结果中区分这些属性,那么应该在SELECT 子句中使用表名或表的别名来限定这些属性。这是因为在没有限定的情况下,如果两个表中有同名的列,查询结果中可能会出现列名冲突,导致结果不明确。通过使用表名或别名限定列名,可以确保查询结果的准确性和可读性。

2) 结果关系中的元组个数

如果两个关系(表)中的元组(行)个数分别是 m 和 n,那么执行交叉连接后得到的结果关系中的元组个数确实是两个关系中的元组个数的乘积,即 mn。这是因为每个来自第一个表的元组都会与第二个表中的每个元组进行组合,生成 mn 个唯一的元组组合。

【例 5.58】 使用交叉连接查询所有农户信息和采购信息。查询结果如图 5.25 所示。代码如下:

```
SELECT *
FROM  FarmerInfo CROSS JOIN ASupplyInfo;
```

下面的语句与上面语句执行结果相同。

```
SELECT *
FROM  FarmerInfo, ASupplyInfo;
```

FPhone	FName	F_BasicInfo	AID	FPhone	Atime	Aquantity
13327**0001	张三丰	主要供应苹果、雪梨等水果	7	13327**0001	2024-01-24 10:30:00	100
13327**0001	张三丰	主要供应苹果、雪梨等水果	6	13927**0088	2024-02-01 15:45:00	30
13327**0001	张三丰	主要供应苹果、雪梨等水果	5	13327**7009	2024-02-01 13:30:00	50
13327**0001	张三丰	主要供应苹果、雪梨等水果	4	13927**6068	2024-02-01 11:15:00	75
13327**0001	张三丰	主要供应苹果、雪梨等水果	3	13927**0088	2024-01-26 14:20:00	50
13327**0001	张三丰	主要供应苹果、雪梨等水果	2	13827**0099	2024-01-25 08:45:00	30
13327**0001	张三丰	主要供应苹果、雪梨等水果	1	13327**0001	2024-01-24 09:00:00	80
13827**3211	张天明	主要供应牛肉	7	13327**0001	2024-01-24 10:30:00	100
13827**3211	张天明	主要供应牛肉	6	13927**0088	2024-02-01 15:45:00	30
13827**3211	张天明	主要供应牛肉	5	13327**7009	2024-02-01 13:30:00	50
13827**3211	张天明	主要供应牛肉	4	13927**6068	2024-02-01 11:15:00	75
13827**3211	张天明	主要供应牛肉	3	13927**0088	2024-01-26 14:20:00	50
13827**3211	张天明	主要供应牛肉	2	13827**0099	2024-01-25 08:45:00	30
13827**3211	张天明	主要供应牛肉	1	13327**0001	2024-01-24 09:00:00	80
13827**0099	李天一	主要供应白牛奶葡萄	7	13327**0001	2024-01-24 10:30:00	100
13827**0099	李天一	主要供应白牛奶葡萄	6	13927**0088	2024-02-01 15:45:00	30
13827**0099	李天一	主要供应白牛奶葡萄	5	13327**7009	2024-02-01 13:30:00	50

图 5.25　交叉连接的查询结果

4. 自连接查询

连接不仅可以在表之间进行,也可以使一个表同其自身进行连接,这种连接称为自连接(SELF JOIN),相应的查询称为自连接查询。在 FROM 子句中可以给这个表取别名,在语句的其他需要使用到该别名的地方用点来连接该别名和字段名。下面举例介绍自连接的应用。

假设有一张行政区划表,其数据表结构如表 5.2 所示。

表 5.2　行政区划表

属 性 名 称	数 据 类 型	长度/字节	PK
区编码	VARCHAR	20	Y
区名称	VARCHAR	30	
上级区编码	VARCHAR	20	

1）使用 CREATE TABLE 语句创建行政区划表

```
CREATE TABLE 行政区划表
(
   区编码   VARCHAR(20) PRIMARY KEY,
   区名称 VARCHAR(30) NOT NULL,
上级区编码   VARCHAR(20)
);
```

2）使用 INSERT 语句添加记录

```
INSERT INTO 行政区划表
  (区编码,区名称,上级区编码)
VALUES
   ('44','广东省','0'),
   ('4401','广州市','44'),
   ('440102','天河区','4401'),
   ('440183','白云区','4401'),
  ('4402','深圳市','44');
```

添加之后的行政区划表的数据如表 5.3 所示。

表 5.3　行政区划表的数据

区　编　码	区　名　称	上级区编码
44	广东省	0
4401	广州市	44
4402	深圳市	44
440102	天河区	4401
440183	白云区	4401

【例 5.59】 使用自连接查询行政区的上下级关系。查询结果如图 5.26 所示。代码如下：

```
SELECT P.区编码   上级编号,P.区名称 上级名称,D.区编码 下级编号,D.区名称 下级名称
FROM   行政区划表 P,行政区划表 D
WHERE   D.上级区编码 = P.区编码;
```



图 5.26　自连接的查询结果

小　　结

本章主要介绍了农产品营销管理数据库的查询方法，即如何使用 SELECT 语句从数据库中查询数据。SELECT 语句是 SQL（结构化查询语言）中最基本且最重要的语句之一，它广泛应用于各种数据库管理系统，包括 MySQL。

本章内容涵盖了单表内的简单数据查询、多表之间的关联查询、各种子句（如 WHERE、GROUP BY、HAVING、ORDER BY 等）的使用、子查询的概念及应用，以及不同类型的连接查询（如左外连接、右外连接）等。此外，还介绍了实用 SQL 语句的使用，旨在帮助读者掌握数据库的高级查询技术。

通过完成本章的学习任务，读者应能够熟练掌握并运用关联表查询、数据统计处理等技术来解决实际问题。这些技能对于开发高效、准确的在线销售助农产品系统至关重要。

动 手 实 践

一、单选题

（1）在查询语句中，给查询结果列重命名可以在原列名的后面加上（　　）关键字。

A. WHERE　　B. GROUP　　C. AS　　D. BY

（2）（　　）子句是对数据按照某个或者多个字段进行分组。

A. WHERE　　B. WITH　　C. GROUP BY　　D. ORDER BY

（3）以下有关空值的叙述中不正确的是（　　）。

A. 用＝NULL 查询指定列为空值的记录

B. 包含空值的表达式其计算结果为空值

C. 聚合函数通常忽略空值

D. 对允许空值的列排序时，包含空值的记录总是排在最前面

（4）SQL 中，与 NOT IN 等价的是（　　）。

A. ＝SOME　　B. <>SOME　　C. ＝ALL　　D. <>ALL

（5）在分组查询中要去掉不满足条件的元组，应当（　　）。

A. 使用 WHERE 子句

B. 在 GROUP BY 中使用 HAVING 子句

C. 先使用 WHERE 子句，后使用 HAVING 子句

D. 先使用 HAVING 子句，后使用 WHERE 子句

（6）SELECT 语句查询条件中的谓词＝ANY 与（　　）等价。

A. IN　　B. NOT IN　　C. BETWEEN　　D. NOT BETWEEN

（7）若要求查找姓李的学生学号和姓名，正确的 SQL 语句是（　　）。

A. SELECT S#，SNAME FROM S WHERE SNAME＝'李%'

B. SELECT S#，SNAME FROM S WHERE SNAME LIKE '李%'

C. SELECT S#，SNAME FROM S WHERE SNAME＝'%李%'

D. SELECT S#，SNAME FROM S WHERE SNAME LIKE '%李%'

(8) 在关系 R 中,若要找出属性 C 的值为'C1'或'C2',并且属性 D 的值大于 10 的所有元组,SELECT 语句中的 WHERE 后应表达为(　　)。

A. C='C1' OR C='C2'AND D>10　B. (C='C1' OR C='C2') AND D>10

C. D>10 AND C='C1' OR C='C2'　D. D>10 OR C='C1' AND C='C2'

(9) 若要找出在关系 R 的 A 属性上不为空的元组,则 SELECT 语句中的 WHERE 后应表达为(　　)。

A. A <> NULL　　B. A <>''

C. A IS NOT NULL　　D. A NOT IS NULL

(10) 已知成绩关系如下表所示,执行下面的 SQL 语句:

```
SELECT COUNT(DISTINCT 学号)    FROM 成绩表    WHERE 分数> 60
```

查询结果是(　　)。

学　　号	课　程　号	分　　数
S1	C1	70
S1	C2	85
S2	C1	NULL
S2	C2	45
S3	C3	90

A. 1　　B. 2　　C. 3　　D. 4

二、填空题

(1) 查询姓名属性中含有"丽"字的条件表达式为:姓名 LIKE _____。

(2) 在学生选课数据库中的三个关系如下:S(SNO,SNAME,SEX,AGE),SC(SNO,CNO,GRADE),C(CNO,CNAME,TEACHER)。现要查找选修"数据库技术"课程的学生姓名和成绩,可使用如下的 SQL 语句:

```
SELECT SNAME,GRADE        FROM S,SC,C
WHERE CNAME = '数据库技术' AND   S.SNO = SC.SNO AND _____。
```

(3) 设有学生关系 STU(SNO,SNAME,AGE,SEX),其属性分别表示学号、姓名、年龄和性别。查询年龄为空值的学生姓名的 SQL 语句是_____。

(4) 设学生表和成绩表的结构分别为(学号,姓名,所在系)和(学号,课程名,成绩),如果希望按分数降序查询出"英语系中选修了计算机课程的学生姓名和成绩",则对应的 SQL 语句是_____。

(5) 已知学生关系(学号,姓名,性别,入学成绩,身高),要查询女生中身高最高的前三个学生的记录信息,则对应的 SQL 语句是_____。

三、综合题

表 1　课程表 Course

列名	数据类型	允许NULL值
Cno	CHAR(4)	☐
Cname	VARCHAR(40)	☐
Cpno	CHAR(4)	☑
Ccredit	SMALLINT	☑

表 2　系部表 Department

列名	数据类型	允许NULL值
Dno	CHAR(2)	☐
Dname	VARCHAR(20)	☑

表 3　选课表 SC

列名	数据类型	允许NULL值
Sno	CHAR(9)	☐
Cno	CHAR(4)	☐
Grade	SMALLINT	☑

表 4　学生表 Student

列名	数据类型	允许NULL值
Sno	CHAR(9)	☐
Sname	VARCHAR(20)	☐
Ssex	CHAR(2)	☐
Sage	SMALLINT	☐
Dno	CHAR(2)	☑

数据库 Stud 存在以上 4 个数据表，以下所有操作都是在数据库 Stud 中进行的，写出相应的 SQL 语句。

(1) 查询年龄不为 15～18 岁的学生的姓名、所在系和年龄。

(2) 查询全体学生的姓名及其出生年份。

(3) 查询姓李的学生数量。

(4) 查询姓张、姓李和姓刘的学生的详细信息。

(5) 查询没有考试成绩的学生的学号和相应的课程号。

(6) 查询选修 1 或 2 号课程且分数大于或等于 85 分的学生的学号、课程号和成绩。

(7) 求学校中共有多少个系。

(8) 找出每个学生超过他自己选修课程的平均成绩的课程号。

(9) 查询每一门课的间接选修课的信息。

(10) 查询选修了 1 号课程的不及格的学生以及缺考的学生。

(11) 查询所有选修了 1 号课程的学生姓名(使用 EXISTS)。

(12) 查询选修了“数学”或“C 语言”课程的学生的学号、姓名、成绩。

(13) 删除学号为 2012622 的学生信息。

(14) 查询每个学生的学号、姓名、选修课的课程名及成绩。

(15) 查询课程成绩低于 75 分的学生的学号、姓名、选修课的课程名及成绩。

(16) 查询与张敏在同一个系学习的学生信息。

(17) 查询其他系中比计算机系某一学生年龄小的学生信息。

(18) 查询系编号及每个系的学生的考试的平均成绩。

(19) 查询选修了“数据库”课程的学生的学号和姓名。

(20) 查询所有学生的学号、姓名、选修课的名称和成绩(没有选修课的同学的选修课信息显示为空)。