

1 检索数据

1. 编写 SQL 语句，从 Customers 表中检索所有的 ID（cust_id）。

```
1 SELECT cust_id
2 FROM customers
```

2. OrderItems 表包含了所有已订购的产品（有些已被订购多次）。编写 SQL 语句，检索并列出已订购产品（prod_id）的清单（不用列每个订单，只列出不同产品的清单）。提示：最终应该显示 7 行。

```
1 SELECT DISTINCT prod_id
2 FROM orderitems
```

3. 编写 SQL 语句，检索 Customers 表中所有的列，再编写另外的 SELECT 语句，仅检索顾客的 ID。使用注释，注释掉一条 SELECT 语句，以便运行另一条 SELECT 语句。（当然，要测试这两个语句。）

```
1 -- SELECT * FROM customers
2 SELECT prod_id
3 FROM customers
```

2 排序检索数据

1. 编写 SQL 语句，从 Customers 中检索所有的顾客名称（cust_names），并按从 Z 到 A 的顺序显示结果。

```
1 SELECT cust_name
2 FROM customers
3 ORDER BY cust_name DESC
```

2. 编写 SQL 语句，从 Orders 表中检索顾客 ID (cust_id) 和订单号 order_num) ，并先按顾客 ID 对结果进行排序，再按订单日期倒序排列。

```
1 SELECT cust_id, order_num
2 FROM orders
3 ORDER BY cust_id, order_num DESC
```

3. 显然，我们的虚拟商店更喜欢出售比较贵的物品，而且这类物品有很多。编写 SQL 语句，显示 OrderItems 表中的数量和价格 (item_price) ，并按数量由多到少、价格由高到低排序。

```
1 SELECT quantity, item_price
2 FROM orderitems
3 ORDER BY quantity DESC, item_price DESC
```

4. 下面的 SQL 语句有问题吗？（尝试在不运行的情况下指出。）

```
SELECT vend_name,
FROM Vendors
ORDER vend_name DESC;
```

第一行去掉', ' ,ORDER后加BY

3 数据过滤

1. 编写 SQL 语句，从 Products 表中检索产品 ID (prod_id) 和产品名称 (prod_name) ，只返回价格为 9.49 美元的产品。

```
1 SELECT prod_id, prod_name
2 FROM products
3 WHERE prod_price = 9.49
```

2. 编写 SQL 语句，从 Products 表中检索产品 ID (prod_id) 和产品名称 (prod_name) ，只返回价格为 9 美元或更高的产品。

```
1 SELECT prod_id, prod_name
2 FROM products
3 WHERE prod_price >= 9
```

3. 结合第 3 课和第 4 课编写 SQL 语句，从 OrderItems 表中检索出所有不同订单号（order_num），其中包含 100 个或更多的产品。

```
1 SELECT DISTINCT order_num
2 FROM orderitems
3 WHERE quantity >= 100
```

4. 编写 SQL 语句，返回 Products 表中所有价格在 3 美元到 6 美元之间的产品的名称（prod_name）和价格（prod_price），然后按价格对结果进行排序。（本题有多种解决方案，我们在下一课再讨论，不过你可以使用目前已学的知识来解决它。）

```
1 SELECT prod_name, prod_price
2 FROM products
3 WHERE prod_price BETWEEN 3 AND 6
```

4 高级数据过滤

1. 编写 SQL 语句，从 Vendors 表中检索供应商名称（vend_name），仅返回加利福尼亚州的供应商（这需要按国家[USA]和州[CA]进行过滤，没准其他国家也存在一个加利福尼亚州）。提示：过滤器需要匹配字符串。

```
1 SELECT order_num, prod_id, quantity
2 FROM orderitems
3 WHERE quantity >=100 AND prod_id IN ('BR01', 'BR02', 'BR03')
```

2. 编写 SQL 语句，查找所有至少订购了总量 100 个的 BR01、BR02 或 BR03 的订单。你需要返回 OrderItems 表的订单号（order_num）、产品 ID（prod_id）和数量，并按产品 ID 和数量进行过滤。提示：根据编写过滤器的方式，可能需要特别注意求值顺序。

```
1 SELECT order_num, prod_id, quantity
2 FROM orderitems
3 WHERE quantity >=100 AND prod_id IN ('BR01', 'BR02',
    'BR03')
```

5 使用通配符进行过滤

1. 编写 SQL 语句，从 Products 表中检索产品名称（prod_name）和描述（prod_desc），仅返回描述中包含 toy 一词的产品。

```
1 SELECT prod_name, prod_desc
2 FROM products
3 WHERE prod_desc like '%toy%'
```

2. 反过来再来一次。编写 SQL 语句，从 Products 表中检索产品名称（prod_name）和描述（prod_desc），仅返回描述中未出现 toy 一词的产品。这次，按产品名称对结果进行排序。

```
1 SELECT prod_name, prod_desc
2 FROM products
3 WHERE prod_desc not like '%toy%'
```

3. 编写 SQL 语句，从 Products 表中检索产品名称（prod_name）和描述（prod_desc），仅返回描述中同时出现 toy 和 carrots 的产品。有好几种方法可以执行此操作，但对于这个挑战题，请使用 AND 和两个 LIKE 比较。

```
1 SELECT prod_name, prod_desc
2 FROM products
3 WHERE prod_desc LIKE '%toy%' AND prod_desc LIKE
   '%carrots'
```

4. 来个比较棘手的。我没有特别向你展示这个语法，而是想看看你根据目前已学的知识是否可以找到答案。编写 SQL 语句，从 Products 表中检索产品名称（prod_name）和描述（prod_desc），仅返回在描述中以先后顺序同时出现 toy 和 carrots 的产品。提示：只需要用带有三个 % 符号的 LIKE 即可。

```
1 SELECT prod_name, prod_desc
2 FROM products
3 WHERE prod_desc LIKE '%toy%carrots%'
```

6 计算字段

1. 别名的常见用法是在检索出的结果中重命名表的列字段（为了符合特定的报表要求或客户需求）。编写 SQL 语句，从 Vendors 表中检索 vend_id、vend_name、vend_address 和 vend_city，将 vend_name 重命名为 vname，将 vend_city 重命名为 vcity，将 vend_address 重命名为 vaddress。按供应商名称对结果进行排序（可以使用原始名称或新的名称）。

```
1 SELECT vend_id, vend_name AS vname, vend_address AS
   vaddress, vend_city AS vcity
2 FROM vendors
3 ORDER BY vname
```

2. 我们的示例商店正在进行打折促销，所有产品均降价 10%。编写 SQL 语句，从 Products 表中返回 prod_id、prod_price 和

sale_price。sale_price 是一个包含促销价格的计算字段。提示：可以乘以 0.9，得到原价的 90%（即 10%的折扣）。

```
1 SELECT prod_id, prod_price, prod_price * 0.9 AS  
   sale_price  
2 FROM products
```

7 使用函数处理数据

1. 我们的商店已经上线了，正在创建顾客账户。所有用户都需要登录名，默认登录名是其名称和所在城市的组合。编写 SQL 语句，返回顾客 ID（cust_id）、顾客名称（customer_name）和登录名（user_login），其中登录名全部为大写字母，并由顾客联系人的前两个字符（cust_contact）和其所在城市的前三个字符（cust_city）组成。例如，我的登录名是 BEOAK（Ben Forta，居住在 Oak Park）。提示：需要使用函数、拼接和别名。

```
1 SELECT cust_id, cust_name,  
   Concat(UPPER(SUBSTR(cust_contact,1,2)),UPPER(SUBSTR(  
   cust_city,1,3))) AS user_login  
2 FROM customers
```

2. 编写 SQL 语句，返回 2020 年 1 月的所有订单的订单号（order_num）和订单日期（order_date），并按订单日期排序。你应该能够根据目前已学的知识来解决此问题，但也可以开卷查阅 DBMS 文档。

```
1 SELECT order_num, order_date  
2 FROM orders  
3 WHERE YEAR(order_date) = '2020' AND  
   MONTH(order_date) = '01'
```

8 汇总数据

1. 编写 SQL 语句，确定已售出产品的总数（使用 OrderItems 中的 quantity 列）。

```
1 SELECT SUM(quantity) AS quantity_num
2 FROM orderitems
```

2. 修改刚刚创建的语句，确定已售出产品项（prod_item）BR01 的总数。

```
1 SELECT SUM(quantity) AS quantity_num
2 FROM orderitems
3 WHERE prod_id = 'BR01'
```

3. 编写 SQL 语句，确定 Products 表中价格不超过 10 美元的最贵产品的价格（prod_price）。将计算所得的字段命名为 max_price。

```
1 SELECT MAX(prod_price) AS max_price
2 FROM products
3 WHERE prod_price <= 10
```

9 分组数据

1. OrderItems 表包含每个订单的每个产品。编写 SQL 语句，返回每个订单号（order_num）各有多少行数（order_lines），并按 order_lines 对结果进行排序。

```
1 SELECT order_num, count(*) AS order_lines
2 FROM orderitems
3 GROUP BY order_num
```

2. 编写 SQL 语句，返回名为 cheapest_item 的字段，该字段包含每个供应商成本最低的产品（使用 Products 表中的 prod_price），然后从最低成本到最高成本对结果进行排序。

```
1 SELECT MIN(prod_price) AS cheapest_item
2 FROM products
3 GROUP BY vend_id
```

3. 确定最佳顾客非常重要，请编写 SQL 语句，返回至少含 100 项的所有订单的订单号（OrderItems 表中的 order_num）。

```
1 SELECT order_num
2 FROM orderitems
3 GROUP BY order_num
4 HAVING sum(quantity) >= 100
```

4. 确定最佳顾客的另一种方式是看他们花了多少钱。编写 SQL 语句，返回总价至少为 1000 的所有订单的订单号（OrderItems 表中的 order_num）。提示：需要计算总和（item_price 乘以 quantity）。按订单号对结果进行排序。

```
1 SELECT order_num
2 FROM orderitems
3 GROUP BY order_num
4 HAVING sum(item_price * quantity) >= 1000
```

5. 下面的 SQL 语句有问题吗？（尝试在不运行的情况下指出。）

```
SELECT order_num, COUNT(*) AS items
FROM OrderItems
GROUP BY items
HAVING COUNT(*) >= 3
ORDER BY items, order_num;
```

应该是GROUP BY order_num

10 使用子查询

1. 使用子查询，返回购买价格为 10 美元或以上产品的顾客列表。你需要使用 OrderItems 表查找匹配的订单号（order_num），然后使用 Order 表检索这些匹配订单的顾客 ID（cust_id）。

```
1 SELECT cust_id
2 FROM orders
3 where order_num in (select order_num
4                     from orderitems
5                     where item_price >= 10)
```

2. 你想知道订购 BR01 产品的日期。编写 SQL 语句，使用子查询来确定哪些订单（在 OrderItems 中）购买了 prod_id 为 BR01 的产品，然后从 Orders 表中返回每个产品对应的顾客 ID（cust_id）和订单日期（order_date）。按订购日期对结果进行排序。

```
1 select cust_id, order_date
2 from orders
3 where order_num in (select order_num
4                     from orderitems
5                     where prod_id = 'BR01')
```

3. 现在我们让它更具挑战性。在上一个挑战题，返回购买 prod_id 为 BR01 的产品的所有顾客的电子邮件（Customers 表中的 cust_email）。提示：这涉及 SELECT 语句，最内层的从 OrderItems 表返回 order_num，中间的从 Customers 表返回 cust_id。

```

1 SELECT cust_email
2 FROM customers
3 WHERE cust_id in (select cust_id
4                     from orders
5                     where order_num in (SELECT
6                                         order_num
7                                         FROM
6                                         orderitems
7                                         WHERE prod_id
= 'BR01'))

```

4. 我们需要一个顾客 ID 列表，其中包含他们已订购的总金额。编写 SQL 语句，返回顾客 ID（Orders 表中的 cust_id），并使用子查询返回 total_ordered 以便返回每个顾客的订单总数。将结果按金额从大到小排序。提示：你之前已经使用 SUM() 计算订单总数。

```

1 SELECT cust_id, SUM(item_price) AS total
2 FROM orderitems, orders
3 WHERE orderitems.order_num = orders.order_num
4 GROUP BY cust_id
5 ORDER BY total DESC

```

5. 再来。编写 SQL 语句，从 Products 表中检索所有的产品名称（prod_name），以及名为 quant_sold 的计算列，其中包含所售产品的总数（在 OrderItems 表上使用子查询和 SUM(quantity) 检索）。

```

1 SELECT prod_name, SUM(quantity) AS quant_sold
2 FROM products, orderitems
3 WHERE products.prod_id = orderitems.prod_id
4 GROUP BY prod_name

```

11 联结表

1. 编写 SQL 语句，返回 Customers 表中的顾客名称（cust_name）和 Orders 表中的相关订单号（order_num），并按顾客名称再按订单号对结果进行排序。实际上是尝试两次，一次使用简单的等联结语法，一次使用 INNER JOIN。

```
1 SELECT cust_name, order_num
2 FROM customers, orders
3 WHERE customers.cust_id = orders.cust_id
4 ORDER BY cust_name, order_num
```

2. 我们来让上一题变得更有用些。除了返回顾客名称和订单号，添加第三列 OrderTotal，其中包含每个订单的总价。有两种方法可执行此操作：使用 OrderItems 表的子查询来创建 OrderTotal 列，或者将 OrderItems 表与现有表联结并使用聚合函数。提示：请注意需要使用完全限定列名的地方。

```
1 SELECT cust_name, OI.order_num, sum(quantity *
   item_price) AS OrderTotal
2 FROM customers AS C, orders AS O, orderitems AS OI
3 WHERE C.cust_id = O.cust_id AND O.order_num =
   OI.order_num
4 GROUP BY order_num
```

3. 我们重新看一下第 11 课的挑战题 2。编写 SQL 语句，检索订购产品 BR01 的日期，这一次使用联结和简单的等联结语法。输出应该与第11 课的输出相同。

```
1 SELECT cust_id, order_date
2 FROM orders, orderitems
3 WHERE orders.order_num = orderitems.order_num AND
   prod_id = 'BR01'
```

4. 很有趣，我们再试一次。重新创建为第 11 课挑战题 3 编写的 SQL 语句，这次使用 ANSI 的 INNER JOIN 语法。在之前编写的代码中使用了两个嵌套的子查询。要重新创建它，需要两个 INNER JOIN 语句，每个语句的格式类似于本课讲到的 INNER JOIN 示例，而且不要忘记 WHERE 子句可以通过 prod_id 进行过滤。

```
1 SELECT cust_email
2 FROM customers, orders, orderitems
3 WHERE customers.cust_id = orders.cust_id AND
   orders.order_num = orderitems.order_num AND prod_id
   = 'BR01'
```

5. 再让事情变得更加有趣些，我们将混合使用联结、聚合函数和分组。准备好了吗？回到第 10 课，当时的挑战是要求查找值等于或大于 1000 的所有订单号。这些结果很有用，但更有用的是订单数量至少达到这个数的顾客名称。因此，编写 SQL 语句，使用联结从 Customers 表返回顾客名称（cust_name），并从 OrderItems 表返回所有订单的总价。提示：要联结这些表，还需要包括 Orders 表（因为 Customers 表与 OrderItems 表不直接相关，Customers 表与 Orders 表相关，而 Orders 表与 OrderItems 表相关）。不要忘记 GROUP BY 和 HAVING，并按顾客名称对结果进行排序。你可以使用简单的等联结或 ANSI 的 INNER JOIN 语法。或者，如果你很勇敢，请尝试使用两种方式编写。

```
1 SELECT cust_name, sum(quantity * item_price) AS
   Total
2 FROM customers, orders, orderitems
3 WHERE customers.cust_id = orders.cust_id AND
   orders.order_num = orderitems.order_num
4 GROUP BY cust_name
5 HAVING sum(item_price * quantity) >= 1000
```

12 创建高级联结

1. 使用 INNER JOIN 编写 SQL 语句，以检索每个顾客的名称（Customers表中的 cust_name）和所有的订单号（Orders 表中的 order_num）。

```
1 SELECT cust_name, order_num
2 FROM customers, orders
3 WHERE customers.cust_id = orders.cust_id
```

2. 修改刚刚创建的 SQL 语句，仅列出所有顾客，即使他们没有下过订单。

```
1 SELECT cust_name, order_num
2 FROM customers LEFT OUTER JOIN Orders ON
3 customers.cust_id = orders.cust_id
```

3. 使用 OUTER JOIN 联结 Products 表和 OrderItems 表，返回产品名称（prod_name）和与之相关的订单号（order_num）的列表，并按商品名称排序。

```
1 SELECT prod_name, order_num
2 FROM products LEFT OUTER JOIN orderitems ON
3 products.prod_id = orderitems.prod_id
4 ORDER BY prod_name
```

4. 修改上一题中创建的 SQL 语句，使其返回每一项产品的总订单数（不是订单号）。

```
1 SELECT prod_name, COUNT(order_num) AS TotalOrders
2 FROM products LEFT OUTER JOIN orderitems ON
3 products.prod_id = orderitems.prod_id
4 GROUP BY prod_name
5 ORDER BY prod_name
```

5. 编写 SQL 语句，列出供应商（Vendors 表中的 vend_id）及其可供产品的数量，包括没有产品的供应商。你需要使用 OUTER JOIN 和 COUNT()聚合函数来计算 Products 表中每种产品的数量。注意：vend_id 列会显示在多个表中，因此在每次引用它时都需要完全限定它。

```
1 SELECT vendors.vend_id, COUNT(prod_id) AS  
  ItemsNumber  
2 FROM products RIGHT OUTER JOIN vendors ON  
3 products.vend_id = vendors.vend_id  
4 GROUP BY vendors.vend_id  
5 ORDER BY ItemsNumber
```

13 组合查询

1. 编写 SQL 语句，将两个 SELECT 语句结合起来，以便从 OrderItems表中检索产品 ID（prod_id）和 quantity。其中，一个 SELECT 语句过滤数量为 100 的行，另一个 SELECT 语句过滤 ID 以 BNBG 开头的产品。按产品 ID 对结果进行排序。

```
1 SELECT prod_id, quantity  
2 FROM orderitems  
3 WHERE quantity = 100  
4 UNION  
5 SELECT prod_id, quantity  
6 FROM orderitems  
7 WHERE prod_id LIKE 'BNBG%'  
8 ORDER BY prod_id
```

2. 重写刚刚创建的 SQL 语句，仅使用单个 SELECT 语句。

```
1 SELECT prod_id, quantity  
2 FROM orderitems  
3 WHERE quantity = 100 OR prod_id LIKE 'BNBG%'  
4 ORDER BY prod_id
```

3. 我知道这有点荒谬，但这节课中的一个注释提到过。编写 SQL 语句，组合 Products 表中的产品名称（prod_name）和 Customers 表中的顾客名称（cust_name）并返回，然后按产品名称对结果进行排序。

```
1 SELECT prod_name
2 FROM products
3 UNION
4 SELECT cust_name
5 FROM customers
6 ORDER BY prod_name
```

4. 下面的 SQL 语句有问题吗？（尝试在不运行的情况下指出。）

```
1 SELECT cust_name, cust_contact, cust_email
2 FROM Customers
3 WHERE cust_state = 'MI'
4 ORDER BY cust_name;
5 UNION
6 SELECT cust_name, cust_contact, cust_email
7 FROM Customers
8 WHERE cust_state = 'IL' ORDER BY cust_name;
```

ORDER BY 语句需要放到最末尾。删除第一处的 ORDER BY cust_name;

14 插入数据

1. 使用 INSERT 和指定的列，将你自己添加到 Customers 表中。明确列出要添加哪几列，且仅需列出你需要的列。

```
1 INSERT INTO
  customers(cust_id,cust_name,cust_city,cust_state,cus
    t_contact,cust_email)
2 VALUE(1, 'Yufo', 'TJ', NULL, 'MySQL', 'Yufo@yufo.com')
```

2. 备份 Orders 表和 OrderItems 表。

```
1 CREATE TABLE order_bak AS SELECT * FROM orders
2 CREATE TABLE orderitems_bak AS SELECT * FROM
  orderitems
```

15 更新和删除数据

1. 美国各州的缩写应始终用大写。编写 SQL 语句来更新所有美国地址，包括供应商状态（Vendors 表中的 vend_state）和顾客状态（Customers 表中的 cust_state），使它们均为大写。

```
1 UPDATE vendors
2 SET vend_state = UPPER(vend_state)
3
4 UPDATE customers
5 SET cust_state = UPPER(vend_state)
```

2. 第 15 课的挑战题 1 要求你将自己添加到 Customers 表中。现在请删除自己。确保使用 WHERE 子句（在 DELETE 中使用它之前，先用 SELECT 对其进行测试），否则你会删除所有顾客！

```
1 DELETE FROM customers
2 WHERE cust_id = 1
```

16 创建和操纵表

1. 在 Vendors 表中添加一个网站列（vend_web）。你需要一个足以容纳 URL 的大文本字段。

```
1 ALTER TABLE vendors
2 ADD vend_web VARCHAR(10000)
```


2. 使用 UPDATE 语句更新 Vendor 记录，以便加入网站（你可以编造任何地址）。

```
1 UPDATE vendors
2 SET vend_web = 'www.15-410.com'
```

17 使用视图

1. 创建一个名为 CustomersWithOrders 的视图，其中包含 Customers 表中的所有列，但仅仅是那些已下订单的列。提示：可以在 Orders 表上使用 JOIN 来仅仅过滤所需的顾客，然后使用 SELECT 来确保拥有正确的数据。

```
1 CREATE VIEW CustomersWithOrders AS
2 SELECT customers.*
3 FROM customers, orders
4 WHERE customers.cust_id = orders.cust_id
5
6 select * from CustomersWithOrders
```

Rows 5								CSV	JSON	CSV	JSON
cust_id	cust_name	cust_address	cust_city	cust_state	cust_zip	cust_country	cust_contact				
1000000001	Village Toys	200 Maple Lane	Detroit	MI	44444	USA	John Smith				
1000000001	Village Toys	200 Maple Lane	Detroit	MI	44444	USA	John Smith				
1000000003	Fun4All	1 Sunny Place	Muncie	IN	42222	USA	Jim Jones				
1000000004	Fun4All	829 Riverside Dr...	Phoenix	AZ	88888	USA	Denise L. Steph				
1000000005	The Toy Store	4545 53rd Street	Chicago	IL	54545	USA	Kim Howard				

2. 下面的 SQL 语句有问题吗？（尝试在不运行的情况下指出。）

```
1 CREATE VIEW OrderItemsExpanded AS
2 SELECT order_num,
3 prod_id,
4 quantity,
5 item_price,
6 quantity*item_price AS expanded_price
7 FROM OrderItems
8 ORDER BY order_num;
```

视图不能使用 `ORDER BY` 语句。（MySQL没有禁止）