Day04-MySQL基础

今日课程学习目标

```
熟悉 SQL语句的执行顺序
常握 不能使用窗口函数的3种情况
常握 GROUP BY和窗口函数配合使用
掌握 GROUP BY分组操作的2个注意点
掌握 COUNT() 计数统计的3个注意点
常握 SQL查询中CASE WHEN语法的使用
```

今日课程内容大纲

```
# 1. 窗口函数避坑指南
    SQL语句的执行顺序
    不能使用窗口函数的情况
        - 情况1：不能在 WHRER 子句中使用窗口函数
        - 情况2：不能在 HAVING 子句中使用窗口函数
        - 情况3：不能在 GROUP BY子句中使用窗口函数
    能够使用窗口函数的情况
        - 可以在SELECT和ORDER BY中使用窗口函数
    GROUP BY和窗口函数配合使用【难点】
# 2. 报表案例
     Northwind 数据集介绍
     SQL数据汇总操作
        -- 多表连接
        -- 带有时间限制的报表
        -- GROUP BY分组操作注意点
        -- COUNT() 计数统计注意点
```

```
17        CASE WHEN 语法
```

知识点1：SQL语句的执行顺序【熟悉】

SQL语句的执行顺序如下：

> FROM > JOIN > ON > WHERE > GROUP BY > 聚合函数 > HAVING > **窗口函数** > SELECT > DISTINCT > ORDER BY > LIMIT

```
1   1）FROM
2   2）JOIN
3   3）ON
4   4）WHERE
5   5）GROUP BY
6   6）聚合函数
7   7）HAVING
8   8）窗口函数
9   9）SELECT
10  10）DISTINCT
11  11）ORDER BY
12  12）LIMIT
```

知识点2：不能使用窗口函数的3种情况【常握】

情况1：不能在 WHRER 子句中使用窗口函数

```
1   -- 需求：查询出所有拍卖中，最终成交价格高于平均成交价格的拍卖
2   -- 查询结果字段：
3   --  id、final_price(最终成交价格)
4
5   # 错误示例
6   SELECT
7       id,
8       final_price
9   FROM auction
10  WHERE final_price > AVG(final_price) OVER();
11
12  -- 子查询
13  SELECT
14      id,
15      final_price
16  FROM auction
17  WHERE final_price > (
18      SELECT
19          AVG(final_price)
20      FROM auction
21  );
```

情况2：不能在 HAVING 子句中使用窗口函数

```
1   -- 需求：查询出国内平均成交价格高于所有拍卖平均成交价格的国家
2   -- 查询结果字段：
3   --  country(国家)、avg(该国家所有拍卖的平均成交价)
4
5   # 错误示例
6   SELECT
7     country,
```

```
  AVG(final_price)  AS `avg`
FROM auction
GROUP BY country
HAVING AVG(final_price) > AVG(final_price) OVER();


# 正确写法(子查询)
SELECT
    country,
    AVG(final_price) AS `avg`
FROM auction
GROUP BY country
HAVING AVG(final_price) > (
    SELECT
        AVG(final_price)
    FROM auction
);
```

情况3：不能在 GROUP BY子句中使用窗口函数

```
-- 需求：将所有的拍卖信息按照浏览次数排序，并均匀分成4组，然后计算每组的最小和最大浏览量
-- 查询结果字段：
--  quartile(分组序号)、min_views(当前组最小浏览量)、max_view(当前组最大浏览量)

# 错误示例
SELECT
  NTILE(4) OVER(ORDER BY views DESC) AS `quartile`,
  MIN(views) AS `min_views`,
  MAX(views) AS `max_views`
FROM auction
GROUP BY NTILE(4) OVER(ORDER BY views DESC);

# 正确实现(子查询)
SELECT
    quartile,
    MIN(views) AS `min_views`,
    MAX(views) AS `max_views`
FROM (
    SELECT
        views,
        NTILE(4) OVER(ORDER BY views DESC) AS `quartile`
    FROM auction
) c
GROUP BY quartile;
```

知识点3：能够使用窗口函数的情况【常握】

> 可以在SELECT和ORDER BY中使用窗口函数

```sql
-- 需求：将所有的拍卖按照浏览量降序排列，并均分成4组，按照每组编号降序排列
-- 查询结果字段：
--  id(拍卖ID)、views(浏览量)、quartile(分组编号)

SELECT
    id,
    views,
    NTILE(4) OVER(ORDER BY views DESC) AS `quartile`
FROM auction
ORDER BY NTILE(4) OVER(ORDER BY views DESC) DESC; -- ORDER BY利用窗口函数的结果列进行排序
```

知识点4：GROUP BY和窗口函数配合使用【掌握】

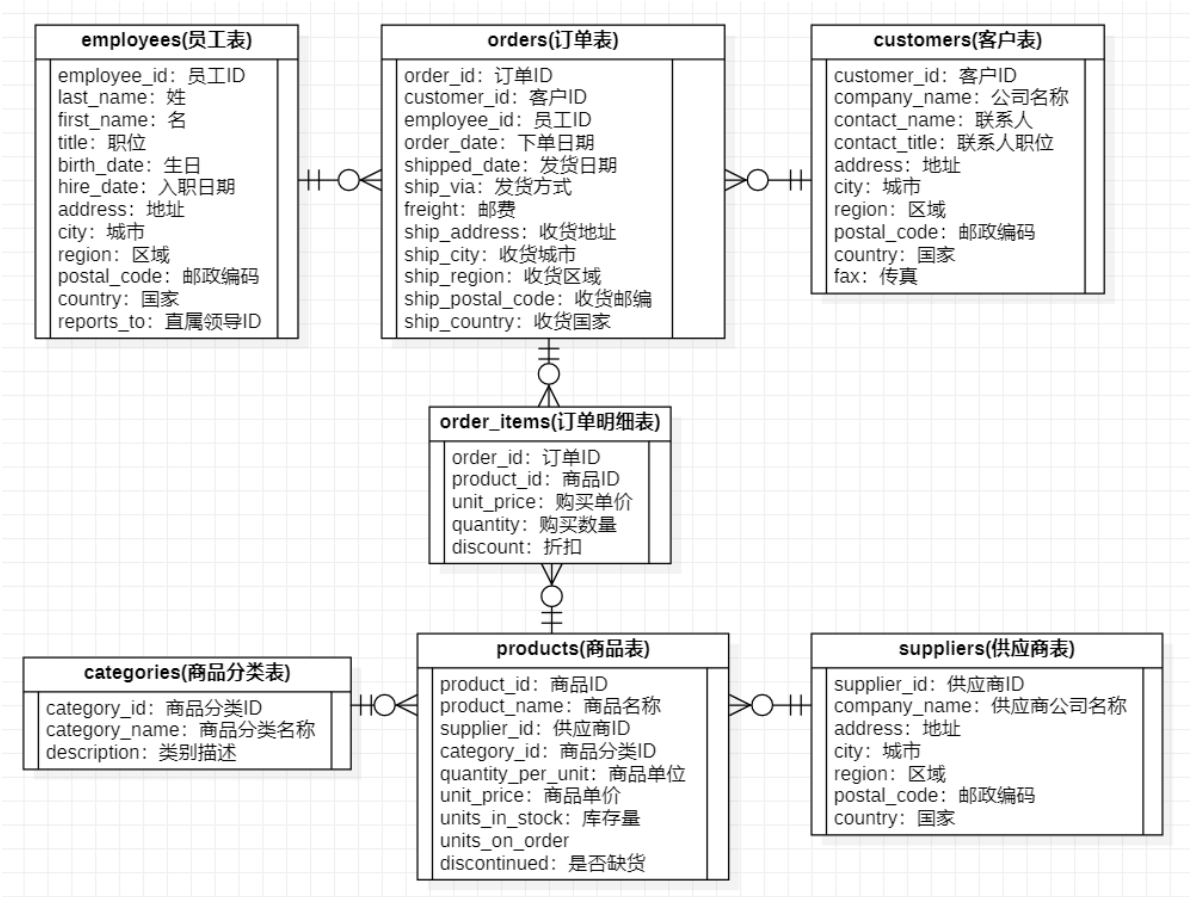
> GROUP BY和窗口函数配合使用时，窗口函数处理的分组聚合之后的结果，不再是原始的表数据。

```sql
-- 练习1
-- 需求：将拍卖数据按国家分组，返回如下信息
-- 查询结果字段：
--  country(国家)、min(每组最少参与人数)、avg(所有组最少参与人数的平均值)


SELECT
    country,
    MIN(participants) AS `min`,
    AVG(MIN(participants)) OVER() AS `avg`
FROM auction
GROUP BY country;

-- 下面的SQL效果等价于上面的SQL
WITH temp_tb AS (
    SELECT
        country,
        MIN(participants) AS `min`
    FROM auction
    GROUP BY country
)
SELECT
    country,
    `min`,
    AVG(`min`) OVER() AS `avg`
FROM temp_tb;

-- 排序函数使用聚合函数的结果
-- 练习2
-- 需求：按国家进行分组，计算了每个国家的拍卖次数，再根据拍卖次数对国家进行排名
-- 查询结果字段：
--  country(国家)、count(该国家的拍卖次数)、rank(按拍卖次数的排名)

SELECT
    country,
    COUNT(id) AS `count`,
    RANK() OVER(ORDER BY COUNT(id) DESC) AS `rank`
FROM auction
GROUP BY country;

```

```
41
42  -- 对GROUP BY分组后的数据使用PARTITION BY
43  -- 练习3
44  -- 需求：将所有的数据按照国家和拍卖结束时间分组，返回如下信息
45  -- 查询结果字段：
46  --  country(国家)、ended(拍卖结束时间)、views_sum(该分组浏览量总和)、
    country_views_sum(分组聚合结果中不同国家拍卖的总浏览量)
47
48
49  SELECT
50      country,
51      ended,
52      SUM(views) AS `views_sum`,
53      SUM(SUM(views)) OVER(PARTITION BY country) AS `country_views_sum`
54  FROM auction
55  GROUP BY country, ended;
```

知识点5：报表案例-Northwind数据集介绍【熟悉】

课程使用微软的 Northwind 数据集, 零售业务，包含了客户，供应商和订单数据。原始数据集可以在 **微软GitHub 仓库** 下载。为了满足课程需求，数据库数据在原始数据基础上做了微调。



知识点6：报表案例-数据集熟悉示例【掌握】

```
1  -- 练习8
2  -- 需求：查询每一个商品的`product_name`、`category_name`、`quantity_per_unit`、
   `unit_price`、`units_in_stock` 并且通过 `unit_price` 字段排序
3
4  SELECT
5      product_name,
6      category_name,
7      quantity_per_unit,
```

```
    unit_price,
    units_in_stock
FROM products p
JOIN categories c
ON p.category_id = c.category_id
ORDER BY unit_price;


-- 练习9
-- 需求：查询提供了3种以上不同商品的供应商列表
-- 查询结果字段：
--     suppler_id(供应商ID)、company_name(供应商公司名称)、products_count(提供的商品数量)

SELECT
    s.supplier_id,
    s.company_name,
    COUNT(*) AS `products_count`
FROM suppliers s
JOIN products p
ON s.supplier_id = p.supplier_id
GROUP BY s.supplier_id, s.company_name
HAVING COUNT(*) > 3;

-- 在标准的 SQL 分组聚合中，除了聚合的结果，其他的列如果在GROUP BY没有出现，不允许出现在SELECT 中
-- 在 MySQL 的分组聚合中，SELECT后面的列值只要在每组内都是唯一的，即使在 GROUP BY 中没有出现，MySQL也不会报错
```

知识点7：SQL数据汇总-多表连接操作【掌握】

```
-- 练习1
-- 需求：查询运输到法国的订单信息，返回如下结果
--
-- 查询结果字段：
-- customer_company_name(客户公司名称)、employee_first_name和employee_last_name(销售员工姓名)、order_date(下单日期)、shipped_date(发货日期)、ship_country(收货国家)

SELECT
    c.company_name AS `customer_company_name`,
    e.first_name AS `employee_first_name`,
    e.last_name AS `employee_last_name`,
    o.order_date,
    o.shipped_date,
    o.ship_country
FROM orders o
JOIN employees e
ON o.employee_id = e.employee_id
JOIN customers c
ON o.customer_id = c.customer_id
WHERE o.ship_country = 'France';


-- 练习2
-- 需求：查询订单编号为10250的订单详情，按商品名称排序，返回如下结果
-- 查询结果字段：
```

```sql
-- product_name(商品名称)、quantity(购买数量)、unit_price(购买单价)、discount(折扣)、order_date(下单日期)

SELECT
    p.product_name,
    oi.quantity,
    oi.unit_price,
    oi.discount,
    o.order_date
FROM orders o
JOIN order_items oi
ON o.order_id = oi.order_id
JOIN products p
ON oi.product_id = p.product_id
WHERE o.order_id = 10250
ORDER BY p.product_name;
```

知识点8：SQL数据汇总-时间筛选vs计算多个对象【掌握】

带有时间限制的报表：

```sql
-- 练习3
-- 需求：统计2016年7月的订单数量
-- 查询结果字段：
--  order_count(2016年7月的订单数量)

SELECT
    COUNT(*) AS `order_count`
FROM orders
WHERE order_date BETWEEN '2016-07-01' AND '2016-07-31';
```

计算多个对象：

```sql
-- 练习4
-- 需求：统计订单号在10200-10260之间的订单中的总商品件数
-- 查询结果字段：
--  order_id(订单ID)、order_items_count(订单中的总商品件数)

SELECT
    order_id,
    COUNT(*) AS `order_items_count`
FROM order_items
WHERE order_id BETWEEN 10200 AND 10260
GROUP BY order_id;
```

知识点9：SQL数据汇总-订单金额计算【掌握】

```sql
-- 练习5
-- 需求：统计ID为10250的订单的总价（折扣前）
-- 查询结果字段：
--  order_id(订单ID)、total_price(订单总价-折扣前)

SELECT
    order_id,
    SUM(unit_price * quantity) AS `total_price`
FROM order_items
WHERE order_id = 10250;

```

```sql

-- 练习6
-- 需求：统计运输到法国的每个订单的总金额
-- 查询结果字段：
--  order_id(订单ID)、company_name(客户公司名称)、total_price(每个订单的总金额)

SELECT
    o.order_id,
    c.company_name,
    SUM(oi.unit_price * oi.quantity) AS `total_price`
FROM orders o
JOIN order_items oi
ON o.order_id = oi.order_id
JOIN customers c
ON o.customer_id = c.customer_id
WHERE o.ship_country = 'France'
GROUP BY o.order_id, c.company_name;
```

知识点10：GROUP BY分组操作的2个注意点【掌握】

注意1：使用GROUP BY分组聚合统计时，需要考虑分组字段中的相同值的业务含义是否相同.

```sql
-- 需求：统计每个员工销售的订单数量
--
-- 查询结果字段：
--  first_name和last_name(员工姓和名)、orders_count(员工销售订单数)

# 注意：这个SQL使用GROUP BY e.first_name, e.last_name分组时，没有考虑员工同名，如果有员工同名，
# 结果会有问题
SELECT
    e.first_name,
    e.last_name,
    COUNT(*) AS `orders_count`
FROM employees e
JOIN orders o
ON e.employee_id = o.employee_id
GROUP BY e.first_name, e.last_name;

# 使用 employee_id 进行分组，每个员工的 employee_id 都是唯一的
SELECT
    e.first_name,
    e.last_name,
    COUNT(*) AS `orders_count`
FROM employees e
JOIN orders o
ON e.employee_id = o.employee_id
GROUP BY e.employee_id, e.first_name, e.last_name;
```

注意2：GROUP BY之后的分组字段不是必须在 SELECT 中出现

```sql
-- 需求：统计2016年6月到2016年7月每个客户的总下单金额，并按金额从高到低排序
--
-- 提示：
--  计算实际总付款金额： SUM(unit_price * quantity * (1 - discount))
--
-- 查询结果字段：
--  company_name(客户公司名称)、total_paid(客户总下单金额-折扣后)
```

```sql

SELECT
    c.company_name,
    SUM(unit_price * quantity * (1 - discount)) AS `total_paid`
FROM customers c
JOIN orders o
ON c.customer_id = o.customer_id
JOIN order_items oi
ON o.order_id = oi.order_id
WHERE o.order_date BETWEEN '2016-06-01' AND '2016-07-31'
GROUP BY c.customer_id, c.company_name
ORDER BY total_paid DESC;
```

知识点11：COUNT()计数操作的3个注意点【掌握】

注意点1：`COUNT(*)` 和 `COUNT(列名)` 之间的区别

- COUNT(*)：进行计数，包括NULL
- COUNT(列名)：对指定列的非NULL数据进行计数

```sql
-- 练习9
-- 需求：统计要发货到不同国家的订单数量以及已经发货的订单数量
--
-- 提示：
--  shipped_date为NULL，表示还未发货
--
-- 查询结果字段：
--  ship_country(国家)、all_orders(总订单数)、shipped_orders(已发货订单数)

SELECT
    ship_country,
    COUNT(*) AS `all_orders`,
    COUNT(shipped_date) AS `shipped_orders`
FROM orders
GROUP BY ship_country;
```

注意点2：`COUNT()` 和 `LEFT JOIN` 配合使用

- 使用SQL出报表时，必须记住关联某些对象可能不存在

```sql
-- 练习10
-- 需求：统计客户ID为 ALFKI、FISSA、PARIS 这三客户各自的订单总数，没有订单的客户也计算在内
--
-- 查询结果字段：
--  customer_id(客户ID)、company_name(客户公司名称)、orders_count(客户订单总数)

SELECT
    c.customer_id,
    c.company_name,
    COUNT(o.order_id) AS `orders_count`
FROM customers c
LEFT JOIN orders o
ON c.customer_id = o.customer_id
WHERE c.customer_id IN ('ALFKI', 'FISSA', 'PARIS')
GROUP BY c.customer_id, c.company_name;
```

注意点3：COUNT()统计时考虑是否需要去重

```
```

```sql
-- 练习11
-- 需求：查询订单运送到西班牙的客户数量
--
-- 提示：
--  一个客户可能下了多个订单
--
-- 查询结果字段：
--  number_of_companies(客户数)


SELECT
    COUNT(DISTINCT customer_id) AS `number_of_companies`
FROM orders
WHERE ship_country = 'Spain';
```

知识点12：CASE WHEN语法简介和基本使用【掌握】

CASE WHEN可以用于SQL查询时，进行条件判断操作。功能类似于Python中的if...elif...else判断。

基础语法：

```sql
CASE
    WHEN 条件1 THEN 值1
    WHEN 条件2 THEN 值2
    WHEN 条件3 THEN 值3
    ...
    ELSE 值n
END
```

查询示例：

```sql
-- 练习1
-- 需求：我们要在报表中显示每种产品的库存量，但我们不想简单地将"units_in_stock"列放在报表中，
-- 还需要按照如下规则显示一个库存级别列：
--
--  库存>100，显示 "high"
--  50 < 库存 <= 100，显示 "moderate"
--  0 < 库存 <= 50，显示 "low"
--  库存=0，显示 "none"
--
-- 查询结果字段：
--  product_id(商品ID)、product_name(商品名称)、units_in_stock(商品库存量)、
stock_level(库存级别)

SELECT
    product_id,
    product_name,
    units_in_stock,
    CASE
        WHEN units_in_stock > 100 THEN 'high'
        WHEN units_in_stock > 50 THEN 'moderate'
        WHEN units_in_stock > 0 THEN 'low'
        WHEN units_in_stock = 0 THEN 'none'
    END AS `stock_level`
FROM products;


-- 练习2
```

```sql
-- 需求：查询客户基本信息报表，返回结果如下：
--
-- 查询结果字段：
--  customer_id(客户ID)、company_name(公司名称)、country(所在国家)、language(使用语言)
--
-- 使用语言的取值规则如下：
--  Germany、Switzerland、and Austria 语言为德语 'German'
--  UK、Canada、the USA、and Ireland 语言为英语 'English'
--  其他所有国家 'Other'

SELECT
    customer_id,
    company_name,
    country,
    CASE
        WHEN country IN ('Germany', 'Switzerland', 'Austria') THEN 'German'
        WHEN country IN ('UK', 'Canada', 'the USA', 'Ireland') THEN 'English'
        ELSE 'Other'
    END AS `language`
FROM customers;
```

知识点13：CASE WHEN配合GROUP BY进行使用【掌握】

```sql
-- 练习4
-- 需求：创建报表统计来自不同大洲的供应商的供应的产品数量(包含未供应产品的供应商)
--
-- 查询结果字段：
--  supplier_continent(大洲)、products_count(供应产品数量)
--
-- 供应商来自哪个大洲的取值规则：
--  `USA`和`Canada`两个国家的大洲取值为：'North America'
--  `Japan`和`Singapore`两个国家的大洲取值为：'Asia'
--  其他国家的大洲取值为 'Other'


-- 标准SQL中，不能使用别名
SELECT
    CASE
        WHEN s.country IN ('USA', 'Canada') THEN 'North America'
        WHEN s.country IN ('Japan', 'Singapore') THEN 'Asia'
        ELSE 'Other'
    END AS `supplier_continent`,
    COUNT(p.product_id) AS `products_count`
FROM suppliers s
LEFT JOIN products p
ON s.supplier_id = p.supplier_id
GROUP BY CASE
        WHEN s.country IN ('USA', 'Canada') THEN 'North America'
        WHEN s.country IN ('Japan', 'Singapore') THEN 'Asia'
        ELSE 'Other'
    END;


-- MySQL中，可以使用别名
SELECT
    CASE
        WHEN s.country IN ('USA', 'Canada') THEN 'North America'
```

```
        WHEN s.country IN ('Japan', 'Singapore') THEN 'Asia'
        ELSE 'Other'
    END AS `supplier_continent`,
    COUNT(p.product_id) AS `products_count`
FROM suppliers s
LEFT JOIN products p
ON s.supplier_id = p.supplier_id
GROUP BY supplier_continent;
```

知识点14：CASE WHEN配合COUNT自定义分组计数【掌握】

可以将 CASE WHEN 和 COUNT 结合使用，自定义分组并统计每组数据数量。

```
-- 练习5
-- 需求：Washington (WA) 是 Northwind 的主要运营地区，统计有多少订单是由华盛顿地区的员工处理的，多少订单是有其他地区的员工处理的
--
-- 查询结果字段：
--  orders_wa_employees(华盛顿地区员工处理订单数)、orders_not_wa_employees(其他地区员工处理订单数)

SELECT
    COUNT(CASE
        WHEN region = 'WA' THEN order_id
    END) AS `orders_wa_employees`,
    COUNT(CASE
        WHEN region != 'WA' THEN order_id
    END) AS `orders_not_wa_employees`
FROM orders o
JOIN employees e
ON o.employee_id = e.employee_id;

-- 练习6
-- 需求：统计运往不同国家的订单中，低运费订单、一般运费订单、高运费订单的数量
--
-- 查询结果字段：
--  ship_country(订单运往国家)、low_freight(低运费订单数量)、moderate_freight(一般运费订单数量)、high_freight(高运费订单数量)

SELECT
    ship_country,
    COUNT(CASE
        WHEN freight < 40.0 THEN order_id
    END) AS `low_freight`,
    COUNT(CASE
        WHEN freight >= 40.0 AND freight < 80.0 THEN order_id
    END) AS `moderate_freight`,
    COUNT(CASE
        WHEN freight >= 80.0 THEN order_id
    END) AS `high_freight`
FROM orders
GROUP BY ship_country;
```

知识点15：CASE WHEN配合SUM进行自定义分组计数【掌握】

上面通过我们通过 COUNT() 函数 和CASE WHEN子句联合使用来创建的报表，也可以通过 SUM() 来替代 COUNT()。

```sql
-- 练习7
-- 需求：Washington (WA) 是 Northwind 的主要运营地区，统计有多少订单是由华盛顿地区的员工处理的，多少订单是有其他地区的员工处理的
--
-- 查询结果字段：
--  orders_wa_employees(华盛顿地区员工处理订单数)、orders_not_wa_employees(其他地区员工处理订单数)

SELECT
    COUNT(CASE
        WHEN region = 'WA' THEN order_id
    END) AS `orders_wa_employees`,
    COUNT(CASE
        WHEN region != 'WA' THEN order_id
    END) AS `orders_not_wa_employees`
FROM orders o
JOIN employees e
ON o.employee_id = e.employee_id;

-- 使用 SUM 来替代 COUNT
SELECT
    SUM(CASE
        WHEN region = 'WA' THEN 1
    END) AS `orders_wa_employees`,
    SUM(CASE
        WHEN region != 'WA' THEN 1
    END) AS `orders_not_wa_employees`
FROM orders o
JOIN employees e
ON o.employee_id = e.employee_id;
```

知识点16：CASE WHEN配合SUM进行复杂计算【掌握】

```sql
-- 练习8
-- 需求：统计每个订单的总金额(折扣后)以及该订单中非素食产品的总金额(折扣后)
--
-- 查询结果字段：
--  order_id(订单ID)、total_price(订单总金额-折扣后)、non_vegetarian_price(订单非素食产品的总金额-折扣后)
--
-- 提示：非素食产品的产品ID （ category_id） 是 6 和 8


SELECT
    o.order_id,
    SUM(quantity * oi.unit_price * (1 - discount)) AS `total_price`,
    SUM(CASE
        WHEN p.category_id IN (6, 8) THEN quantity * oi.unit_price * (1 - discount)
        ELSE 0
    END) AS `non_vegetarian_price`
FROM orders o
```

```sql
JOIN order_items oi
ON o.order_id = oi.order_id
JOIN products p
ON oi.product_id = p.product_id
GROUP BY o.order_id;


-- 练习9
-- 需求：制作报表统计所有订单的总价(折扣前)，并将订单按总价分成3类：high、average、low
--
-- 查询结果字段：
--  order_id(订单ID)、total_price(订单总金额)、price_group(订单总金额分类)
--
-- 订单总金额分类规则：
--  总价超过2000美元：'high'
--  总价在600到2000美元之间：'average'
--  总价低于600美元：'low'


SELECT
    o.order_id,
    SUM(oi.unit_price * oi.quantity) AS `total_price`,
    CASE
        WHEN SUM(oi.unit_price * oi.quantity) > 2000 THEN 'high'
        WHEN SUM(oi.unit_price * oi.quantity) > 600 THEN 'average'
        ELSE 'low'
    END AS `price_group`
FROM orders o
JOIN order_items oi
ON o.order_id = oi.order_id
GROUP BY o.order_id;
```