Hive数据仓库综合实验

已知sales_info数据库中有三张表（`sales_info_raw`、`customers`、`products`）设计的 Hive 实验题目

启动环境，选择数据库

```
use sales_info;
```

完成下面的实验

一、基础查询类实验

1. SELECT 实验

实验1：查询所有订单编号与对应销售地区

```
SELECT order_id, region FROM sales_info_raw;
```

实验2：查询所有客户的姓名与年龄

```
SELECT customer_name, customer_age FROM customers;
```

实验3：查询所有商品的名称和类别

```
SELECT product_name, category FROM products;
```

---

2. DISTINCT 实验

实验1：查询不同的支付方式

```
SELECT DISTINCT payment_method FROM sales_info_raw;
```

实验2：查询销售表中出现过的不同客户ID

```
SELECT DISTINCT customer_id FROM sales_info_raw;
```

实验3：查询客户表中所有不同年龄段

```
SELECT DISTINCT customer_age FROM customers;
```

---

3. WHERE 实验

实验1：查询购买数量大于5的销售记录

```sql
SELECT * FROM sales_info_raw WHERE quantity > 5;
```

实验2：查询年龄小于30岁的客户信息

```sql
SELECT * FROM customers WHERE customer_age < 30;
```

实验3：查询北京地区的订单

```sql
SELECT * FROM sales_info_raw WHERE region = '北京';
```

---

4. GROUP BY 实验

实验1：统计每种支付方式的订单数量

```sql
SELECT payment_method, COUNT(*) AS order_count
FROM sales_info_raw
GROUP BY payment_method;
```

实验2：统计每个地区销售的总数量

```sql
SELECT region, SUM(quantity) AS total_quantity
FROM sales_info_raw
GROUP BY region;
```

实验3：统计每个客户的购买总金额

```sql
SELECT customer_id, SUM(quantity * unit_price) AS total_spent
FROM sales_info_raw
GROUP BY customer_id;
```

---

5. HAVING 实验

实验1：筛选总购买金额大于1000的客户

```sql
SELECT customer_id, SUM(quantity * unit_price) AS total_spent
FROM sales_info_raw
GROUP BY customer_id
HAVING total_spent > 1000;
```

实验2：筛选销售总量超过20的产品

```sql
SELECT product_id, SUM(quantity) AS total_quantity
FROM sales_info_raw
GROUP BY product_id
HAVING total_quantity > 20;
```

实验3：筛选订单数量多于3种的支付方式

```sql
SELECT payment_method, COUNT(*) AS pay_count
FROM sales_info_raw
GROUP BY payment_method
HAVING pay_count > 3;
```

---

6. LIMIT 实验

实验1：查询前5条销售记录

```sql
SELECT * FROM sales_info_raw LIMIT 5;
```

实验2：查询年龄最小的5位客户

```sql
SELECT * FROM customers ORDER BY customer_age ASC LIMIT 5;
```

实验3：查询销售金额前10的记录

```sql
SELECT *, quantity * unit_price AS total
FROM sales_info_raw
ORDER BY total DESC
LIMIT 10;
```

二、高阶查询类实验

7. ORDER BY 实验

实验1：按销售日期升序查询销售记录

```sql
SELECT * FROM sales_info_raw ORDER BY sale_date ASC;
```

实验2：按客户年龄降序查询客户信息

```sql
SELECT * FROM customers ORDER BY customer_age DESC;
```

实验3：按商品类别排序

```sql
SELECT * FROM products ORDER BY category;
```

---

8. CLUSTER BY 实验

实验1：按销售地区进行分桶排序

```
SELECT * FROM sales_info_raw CLUSTER BY region;
```

实验2：按客户性别分桶

```
SELECT * FROM customers CLUSTER BY customer_gender;
```

实验3：按销售数量分桶

```
SELECT * FROM sales_info_raw CLUSTER BY quantity;
```

---

9. DISTRIBUTE BY + SORT BY 实验

实验1：按地区分发数据，并在每个分区中按数量升序排序

```
SELECT * FROM sales_info_raw
DISTRIBUTE BY region
SORT BY quantity ASC;
```

实验2：按客户性别分发数据，年龄降序排序

```
SELECT * FROM customers
DISTRIBUTE BY customer_gender
SORT BY customer_age DESC;
```

实验3：按商品类别分区、名称排序

```
SELECT * FROM products
DISTRIBUTE BY category
SORT BY product_name;
```

三、子查询与函数类实验

10. FROM 子句中的子查询

实验1：查询每个客户的总金额，再筛选金额大于1000的记录

```
SELECT * FROM (
  SELECT customer_id, SUM(quantity * unit_price) AS total_spent
  FROM sales_info_raw
  GROUP BY customer_id
) t WHERE total_spent > 1000;
```

实验2：查询每个支付方式的订单数，并显示排序后的前3名

```sql
SELECT * FROM (
  SELECT payment_method, COUNT(*) AS cnt
  FROM sales_info_raw
  GROUP BY payment_method
) t ORDER BY cnt DESC LIMIT 3;
```

实验3：查询销量前5的商品名称和类别

```sql
SELECT p.product_name, p.category
FROM (
  SELECT product_id, SUM(quantity) AS total_q
  FROM sales_info_raw
  GROUP BY product_id
  ORDER BY total_q DESC
  LIMIT 5
) top5
JOIN products p ON top5.product_id = p.product_id;
```

---

11. WHERE 子句中的子查询

实验1：查询那些下过订单的客户

```sql
SELECT * FROM customers
WHERE customer_id IN (
  SELECT DISTINCT customer_id FROM sales_info_raw
);
```

实验2：查询销售表中销售的产品中类别为"电子产品"的记录

```sql
SELECT * FROM sales_info_raw
WHERE product_id IN (
  SELECT product_id FROM products WHERE category = '电子产品'
);
```

实验3：查询北京地区购买过"华为手机"的订单

```sql
SELECT * FROM sales_info_raw
WHERE region = '北京'
AND product_id IN (
  SELECT product_id FROM products WHERE product_name = '华为手机'
);
```

---

12. WITH 子句实验

实验1：使用 WITH 查询每个客户的总消费金额

```sql
WITH customer_total AS (
  SELECT customer_id, SUM(quantity * unit_price) AS total_spent
  FROM sales_info_raw
  GROUP BY customer_id
)
SELECT * FROM customer_total;
```

实验2：使用 WITH 查询每个商品的总销量，并按降序排列

```sql
WITH product_sales AS (
  SELECT product_id, SUM(quantity) AS total_quantity
  FROM sales_info_raw
  GROUP BY product_id
)
SELECT * FROM product_sales ORDER BY total_quantity DESC;
```

实验3：使用 WITH 查询每种支付方式的订单数

```sql
WITH pay_count AS (
  SELECT payment_method, COUNT(*) AS pay_total
  FROM sales_info_raw
  GROUP BY payment_method
)
SELECT * FROM pay_count;
```

---

13. JOIN 实验

实验1：查询订单明细中包含客户姓名的信息

```sql
SELECT s.*, c.customer_name
FROM sales_info_raw s
inner JOIN customers c ON s.customer_id = c.customer_id;
```

实验2：查询订单明细中包含产品名称的信息

```sql
SELECT s.*, p.product_name
FROM sales_info_raw s
inner JOIN products p ON s.product_id = p.product_id;
```

实验3：查询客户姓名、购买商品名及数量

```sql
SELECT c.customer_name, p.product_name, s.quantity
FROM sales_info_raw s
inner JOIN customers c ON s.customer_id = c.customer_id
inner JOIN products p ON s.product_id = p.product_id;
```

好的，以下是基于你提供的表结构（`sales_info_raw`、`customers`、`products`）设计的 **3个 Hive 左连接（LEFT JOIN）实验题目**，每个实验都附有简要描述和对应的 SQL 代码，适合课堂教学与练习使用。

实验4：查询所有销售记录及对应客户姓名（即便客户信息缺失）

目的： 保留所有订单，即使客户表中没有该客户信息。

```
SELECT s.order_id, s.customer_id, c.customer_name
FROM sales_info_raw s
LEFT JOIN customers c ON s.customer_id = c.customer_id;
```

实验5：查询所有销售记录及对应产品名称和类别（保留没有匹配产品的订单）

目的： 查看是否有无效 product_id 的订单。

```
SELECT s.order_id, s.product_id, p.product_name, p.category
FROM sales_info_raw s
LEFT JOIN products p ON s.product_id = p.product_id;
```

实验6：统计每位客户的订单总数（包含从未下单的客户）

目的： 从客户角度出发，统计订单数量，保留从未下单的客户记录。

```
SELECT c.customer_id, c.customer_name, COUNT(s.order_id) AS order_count
FROM customers c
LEFT JOIN sales_info_raw s ON c.customer_id = s.customer_id
GROUP BY c.customer_id, c.customer_name;
```

四、函数类实验

14.内置函数实验

字符串函数

实验1：将客户姓名全部转为大写

```
SELECT customer_name, UPPER(customer_name) AS upper_name
FROM customers;
```

实验2：获取产品类别的前两个字符

```
SELECT product_name, category, SUBSTRING(category, 1, 2) AS short_cat
FROM products;
```

实验3：拼接客户ID与客户名

```
SELECT CONCAT(customer_id, '-', customer_name) AS full_info
FROM customers;
```

数学函数

实验1：计算每笔订单的总金额，并向上取整

```
SELECT order_id, quantity * unit_price AS total,
       CEIL(quantity * unit_price) AS total_ceiled
FROM sales_info_raw;
```

实验2：对每个订单金额取平方根

```
SELECT order_id, SQRT(quantity * unit_price) AS root_amount
FROM sales_info_raw;
```

日期函数

实验1：提取销售日期中的月份

```
SELECT sale_date, MONTH(sale_date) AS sale_month
FROM sales_info_raw;
```

实验2：格式化销售日期为 "yyyy年MM月dd日"

```
SELECT sale_date, DATE_FORMAT(sale_date, 'yyyy年MM月dd日') AS formatted_date
FROM sales_info_raw;
```

实验3：计算销售日期距离当前日期的天数

```
SELECT sale_date, DATEDIFF(CURRENT_DATE, sale_date) AS days_ago
FROM sales_info_raw;
```

条件函数

实验1：根据客户年龄划分年龄段

```
SELECT customer_name, customer_age,
       CASE
           WHEN customer_age < 25 THEN '青年'
           WHEN customer_age >= 25 AND customer_age < 45 THEN '中年'
           ELSE '老年'
       END AS age_group
FROM customers;
```

实验2：标记是否大额订单（金额超过1000）

```
SELECT order_id, quantity * unit_price AS total,
       IF(quantity * unit_price > 1000, '大额订单', '普通订单') AS order_level
FROM sales_info_raw;
```

聚合函数

实验1：统计每种支付方式的订单数量

```
SELECT payment_method, COUNT(*) AS order_count
FROM sales_info_raw
GROUP BY payment_method;
```

实验2：统计每位客户的总消费金额

```
SELECT customer_id, SUM(quantity * unit_price) AS total_spent
FROM sales_info_raw
GROUP BY customer_id;
```

实验3：查询所有订单的最大单笔金额

```
SELECT MAX(quantity * unit_price) AS max_order_amount
FROM sales_info_raw;
```

类型转换函数

实验1：将销售日期转为字符串格式

```
SELECT sale_date, CAST(sale_date AS STRING) AS sale_date_str
FROM sales_info_raw;
```

实验2：将单价字段转换为 DOUBLE 类型

```
SELECT unit_price, CAST(unit_price AS DOUBLE) AS price_double
FROM sales_info_raw;
```

15. 窗口函数实验

实验1：每个地区订单按金额降序排名（RANK）

```
SELECT *,
       RANK() OVER (PARTITION BY region ORDER BY quantity * unit_price DESC) AS
rnk
FROM sales_info_raw;
```

实验2：每个客户的最近一条订单（ROW_NUMBER）

```
SELECT * FROM (
  SELECT *, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY sale_date DESC)
AS rn
  FROM sales_info_raw
) t WHERE rn = 1;
```

实验3：每种产品的销售量累计和（SUM OVER）

```sql
SELECT product_id, quantity,
       SUM(quantity) OVER (PARTITION BY product_id ORDER BY sale_date) AS
running_total
FROM sales_info_raw;
```

五、综合实验

实验1：查询每位客户的总消费金额和订单数，并显示客户姓名

知识点：`JOIN + GROUP BY + 聚合函数`

```sql
SELECT c.customer_name,
       COUNT(s.order_id) AS order_count,
       SUM(s.quantity * s.unit_price) AS total_spent
FROM sales_info_raw s
LEFT JOIN customers c ON s.customer_id = c.customer_id
GROUP BY c.customer_name;
```

实验2：统计每个地区的总销售金额，并筛选出大于500000的地区

知识点：`GROUP BY + HAVING + 计算字段`

```sql
SELECT region,
       SUM(quantity * unit_price) AS total_sales
FROM sales_info_raw
GROUP BY region
HAVING total_sales > 500000;
```

实验3：查询购买了“电子产品”类商品的客户姓名和购买日期

知识点：`JOIN + 子查询 + 条件查询`

```sql
SELECT DISTINCT c.customer_name, s.sale_date
FROM sales_info_raw s
JOIN customers c ON s.customer_id = c.customer_id
WHERE s.product_id IN (
  SELECT product_id FROM products WHERE category = '电子产品'
);
```

实验4：使用 WITH 查询不同产品的总销售量，并按降序排序

知识点：`WITH + 聚合 + 排序`

```sql
WITH product_sales AS (
  SELECT product_id, SUM(quantity) AS total_quantity
  FROM sales_info_raw
  GROUP BY product_id
)
SELECT p.product_name, ps.total_quantity
FROM product_sales ps
JOIN products p ON ps.product_id = p.product_id
ORDER BY ps.total_quantity DESC;
```

实验5：统计不同支付方式下的平均订单金额和最大金额

知识点：`GROUP BY + 聚合函数（AVG、MAX）`

```sql
SELECT payment_method,
       AVG(quantity * unit_price) AS avg_amount,
       MAX(quantity * unit_price) AS max_amount
FROM sales_info_raw
GROUP BY payment_method;
```

实验6：查询每位客户最近一笔订单的详细信息

知识点：`窗口函数 ROW_NUMBER + 子查询`

```sql
SELECT *
FROM (
  SELECT s.*, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY sale_date
DESC) AS rn
  FROM sales_info_raw s
) t
WHERE rn = 1;
```

实验7：按客户性别统计人均消费金额（包含0元的未消费客户）

知识点：`LEFT JOIN + GROUP BY + 聚合 + NULL处理`

```sql
SELECT c.customer_gender,
       ROUND(SUM(COALESCE(s.quantity * s.unit_price, 0)) / COUNT(DISTINCT
c.customer_id), 2) AS avg_spent_per_customer
FROM customers c
LEFT JOIN sales_info_raw s ON c.customer_id = s.customer_id
GROUP BY c.customer_gender;
```

实验8：将所有销售记录按地区分区，且每区显示前两名订单金额最大的订单

知识点：`窗口函数 RANK + 分区排序`

```sql
SELECT *
FROM (
  SELECT *,
         RANK() OVER (PARTITION BY region ORDER BY quantity * unit_price DESC) AS
rk
  FROM sales_info_raw
) t
WHERE rk <= 2;
```

实验9：计算每个订单的消费金额及其在所在支付方式内的消费排名

知识点：`窗口函数 RANK + 计算字段`

```sql
SELECT order_id,
       payment_method,
       quantity * unit_price AS order_amount,
       RANK() OVER (PARTITION BY payment_method ORDER BY quantity * unit_price
DESC) AS rank_in_method
FROM sales_info_raw;
```

实验10：查询客户姓名、订单数量、消费总额，并按消费金额倒序排序，限制前10名

知识点：`JOIN + GROUP BY + ORDER BY + LIMIT`

```sql
SELECT c.customer_name,
       COUNT(s.order_id) AS order_count,
       SUM(s.quantity * s.unit_price) AS total_spent
FROM sales_info_raw s
JOIN customers c ON s.customer_id = c.customer_id
GROUP BY c.customer_name
ORDER BY total_spent DESC
LIMIT 10;
```

实验11：查询30岁以下客户的订单详情，包括商品名称与支付方式

知识点：`JOIN + WHERE + 条件筛选`

```sql
SELECT c.customer_name, c.customer_age, p.product_name, s.payment_method
FROM sales_info_raw s
JOIN customers c ON s.customer_id = c.customer_id
JOIN products p ON s.product_id = p.product_id
WHERE c.customer_age < 30;
```

实验12：查询每类商品在每个地区的销售总金额

知识点：`JOIN + GROUP BY + 聚合`

```sql
SELECT p.category, s.region, SUM(s.quantity * s.unit_price) AS total_sales
FROM sales_info_raw s
JOIN products p ON s.product_id = p.product_id
GROUP BY p.category, s.region;
```

实验13：统计每月的销售订单数量

知识点：`日期函数 + GROUP BY`

```sql
SELECT DATE_FORMAT(sale_date, 'yyyy-MM') AS month,
       COUNT(*) AS order_count
FROM sales_info_raw
GROUP BY DATE_FORMAT(sale_date, 'yyyy-MM');
```

实验14：找出订单总金额超过平均值的订单

知识点：`聚合函数 + 子查询 + 计算字段`

```sql
SELECT *
FROM sales_info_raw
WHERE quantity * unit_price > (
    SELECT AVG(quantity * unit_price) FROM sales_info_raw
);
```

实验15：查询每种支付方式下最贵的订单（金额最高）

知识点：`窗口函数 RANK + PARTITION BY`

```sql
SELECT *
FROM (
  SELECT *, RANK() OVER (PARTITION BY payment_method ORDER BY quantity *
unit_price DESC) AS rnk
  FROM sales_info_raw
) t
WHERE rnk = 1;
```

实验16：将每位客户的姓名与性别合并为一个新字段“基本信息”

知识点：`字符串函数 CONCAT`

```sql
SELECT customer_id,
       CONCAT(customer_name, '（', customer_gender, '）') AS base_info
FROM customers;
```

实验17：找出未购买过“华为手机”的客户姓名

知识点：`NOT IN + 子查询`

```sql
SELECT customer_name
FROM customers
WHERE customer_id NOT IN (
  SELECT DISTINCT s.customer_id
  FROM sales_info_raw s
  JOIN products p ON s.product_id = p.product_id
  WHERE p.product_name = '华为手机'
);
```

实验18：计算每位客户订单的平均金额，结果保留两位小数

知识点：`GROUP BY + 聚合函数 + ROUND`

```sql
SELECT customer_id,
       ROUND(SUM(quantity * unit_price) / COUNT(order_id), 2) AS avg_order_amount
FROM sales_info_raw
GROUP BY customer_id;
```

实验19：查询所有客户及其是否下过订单（有/无）

知识点：`LEFT JOIN + CASE WHEN`

```sql
SELECT c.customer_name,
       CASE
           WHEN s.order_id IS NOT NULL THEN '有订单'
           ELSE '无订单'
       END AS order_status
FROM customers c
LEFT JOIN sales_info_raw s ON c.customer_id = s.customer_id;
```

实验20：统计每个年龄段（青年、中年、老年）客户的总消费金额

知识点：`CASE WHEN + GROUP BY + 聚合函数`

```sql
SELECT
  CASE
    WHEN customer_age < 30 THEN '青年'
    WHEN customer_age >= 30 AND customer_age < 50 THEN '中年'
    ELSE '老年'
  END AS age_group,
  SUM(s.quantity * s.unit_price) AS total_spent
FROM customers c
JOIN sales_info_raw s ON c.customer_id = s.customer_id
GROUP BY
  CASE
    WHEN customer_age < 30 THEN '青年'
    WHEN customer_age >= 30 AND customer_age < 50 THEN '中年'
    ELSE '老年'
  END;
```
