Hive 제 5 장 DML 연산 실습 워크북 — 연습문제 참고 정답

> ⚠️ **교사 전용 문서**: 수업 전 학생에게 배포하지 마세요
> **대응 워크북**: Hive 제 5 장 DML 연산 실습 워크북
> **언어**: 한국어

제 1 장 LOAD DATA 연습문제 정답

연습 1-1 참고 정답

문제: `OVERWRITE` 방식으로 `employees.txt`를 `employees` 테이블에 다시 로드하여 테이블에 정확히 15 개의 레코드만 남도록 하세요.

```sql
USE unit5db;

-- 단계 1: OVERWRITE 방식으로 로드하여 기존 데이터 덮어쓰기
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/employees.txt'
OVERWRITE INTO TABLE employees;

-- 단계 2: 데이터 건수가 15 인지 확인
SELECT COUNT(*) AS total FROM employees;
```

예상 출력:

```
+--------+
| total  |
+--------+
| 15     |
+--------+
```

해설: `OVERWRITE` 키워드를 사용하면 기존 데이터를 먼저 삭제한 후 새 파일을 가져오므로,

Hive 第 5 章 DML 操作实践工作手册 — 练习题参考答案

> ⚠️ **教师专用文档**：请勿在课前发放给学生

> **对应工作手册**：Hive 第 5 章 DML 操作实践工作手册
> **语言**：中文

第 1 章 LOAD DATA 练习题答案

练习 1-1 参考答案

题目：请重新用 OVERWRITE 方式加载 employees.txt 到 employees 表中，确保表中只有 15 条记录。

```sql
USE unit5db;

-- 步骤 1：使用 OVERWRITE 方式加载，覆盖旧数据
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/employees.txt'
OVERWRITE INTO TABLE employees;

-- 步骤 2：验证数据条数应为 15
SELECT COUNT(*) AS total FROM employees;
```

预期输出：

```
+--------+
| total  |
+--------+
| 15     |
+--------+
```

解析：使用 `OVERWRITE` 关键字会先清空表中现有数据，再导入新文件，因此最终只有文件中的

최종적으로 파일의 15 개 레코드만 남아 중복이 발생하지 않습니다.

연습 1-2 참고 정답

문제: `sales` 테이블의 데이터 건수를 확인하고, 판매 금액이 가장 큰 상위 3 건의 레코드를 조회하세요.

```sql
-- 단계 1: sales 테이블 데이터 건수 집계
SELECT COUNT(*) AS total_sales FROM sales;

-- 단계 2: 판매 금액이 가장 큰 상위 3 건 조회
SELECT
    sale_id,
    emp_id,
    sale_date,
    amount,
    product_type,
    region
FROM sales
ORDER BY amount DESC
LIMIT 3;
```

예상 출력 (건수):

```
+-------------+
| total_sales |
+-------------+
| 20          |
+-------------+
```

예상 출력 (상위 3 건):

```
+---------+--------+------------+---------+--------------+--------+
| sale_id | emp_id | sale_date  | amount  | product_type | region |
+---------+--------+------------+---------+--------------+--------+
| S017    | E015   | 2023-02-08 | 200000  | 服务项目     | 华南   |
| S019    | E015   | 2023-03-12 | 185000  | 软件产品     | 华南   |
| S018    | E006   | 2023-02-25 | 168000  | 硬件设备     | 华南   |
```

15 条记录，不会发生重复。

练习 1-2 参考答案

题目：验证 sales 表中的数据条数，并查看销售金额最大的 3 条记录。

```sql
-- 步骤 1：统计 sales 表数据条数
SELECT COUNT(*) AS total_sales FROM sales;

-- 步骤 2：查看销售金额最大的 3 条记录
SELECT
    sale_id,
    emp_id,
    sale_date,
    amount,
    product_type,
    region
FROM sales
ORDER BY amount DESC
LIMIT 3;
```

预期输出（条数）：

```
+-------------+
| total_sales |
+-------------+
| 20          |
+-------------+
```

预期输出（Top 3）：

```
+---------+--------+------------+---------+--------------+--------+
| sale_id | emp_id | sale_date  | amount  | product_type | region |
+---------+--------+------------+---------+--------------+--------+
| S017    | E015   | 2023-02-08 | 200000  | 服务项目     | 华南   |
| S019    | E015   | 2023-03-12 | 185000  | 软件产品     | 华南   |
| S018    | E006   | 2023-02-25 | 168000  | 硬件设备     | 华南   |
```

```
+---------+--------+-----------+---------+-------
-------+--------+
```

연습 1-3 참고 정답

문제: `dept_backup` 테이블을 생성하고 `departments.txt` 데이터를 로드하세요.

```sql
-- 단계 1: departments 와 동일한 구조의 백업 테이블 생성
CREATE TABLE IF NOT EXISTS dept_backup (
    dept_id    STRING   COMMENT '부서 번호',
    dept_name  STRING   COMMENT '부서명',
    location   STRING   COMMENT '소재 도시',
    budget     BIGINT   COMMENT '부서 예산 (원)'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 단계 2: 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
INTO TABLE dept_backup;

-- 단계 3: 데이터 확인
SELECT * FROM dept_backup;
SELECT COUNT(*) AS total FROM dept_backup;
```

예상 출력 (총 6 행):

```
+----------+------------+--------+----------+
| dept_id  | dept_name  |location|  budget  |
+----------+------------+--------+----------+
| D001     | 技术研发部  | 北京   | 1500000  |
| D002     | 市场营销部  | 上海   | 800000   |
| D003     | 人力资源部  | 北京   | 400000   |
| D004     | 财务管理部  | 广州   | 600000   |
| D005     | 客户服务部  | 深圳   | 500000   |
| D006     | 产品设计部  | 杭州   | 700000   |
+----------+------------+--------+----------+
```

练习 1-3 参考答案

题目：创建 dept_backup 表，加载 departments.txt 数据。

```sql
-- 步骤 1：创建与 departments 相同结构的备份表

CREATE TABLE IF NOT EXISTS dept_backup (
    dept_id    STRING   COMMENT '部门编号',
    dept_name  STRING   COMMENT '部门名称',
    location   STRING   COMMENT '所在城市',
    budget     BIGINT   COMMENT '部门预算（元）'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 步骤 2：加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
INTO TABLE dept_backup;

-- 步骤 3：验证数据
SELECT * FROM dept_backup;
SELECT COUNT(*) AS total FROM dept_backup;
```

预期输出（共 6 行）：

```
+----------+------------+--------+----------+
| dept_id  | dept_name  |location|  budget  |
+----------+------------+--------+----------+
| D001     | 技术研发部  | 北京   | 1500000  |
| D002     | 市场营销部  | 上海   | 800000   |
| D003     | 人力资源部  | 北京   | 400000   |
| D004     | 财务管理部  | 广州   | 600000   |
| D005     | 客户服务部  | 深圳   | 500000   |
| D006     | 产品设计部  | 杭州   | 700000   |
+----------+------------+--------+----------+
```

제 2 장 INSERT 연습문제 정답

연습 2-1 참고 정답

문제: `emp_test` 테이블에 자신의 정보를 삽입하세요（직원 번호 T999）.

```sql
USE unit5db;

-- emp_test 테이블 존재 여부 확인
SHOW TABLES LIKE 'emp_test';

-- 없으면 먼저 생성
CREATE TABLE IF NOT EXISTS emp_test (
    emp_id    STRING,
    emp_name  STRING,
    salary    DOUBLE,
    dept      STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 데이터 삽입 (예시, 학생이 자신의 정보로 작성)
INSERT INTO TABLE emp_test
VALUES ('T999', 'ZhangSan', 12000, '大数据技术部');

-- 삽입 결과 확인
SELECT * FROM emp_test WHERE emp_id = 'T999';
```

예상 출력:

```
+--------+----------+--------+--------------+
| emp_id | emp_name | salary |     dept     |
+--------+----------+--------+--------------+
| T999   | ZhangSan | 12000  | 大数据技术部  |
+--------+----------+--------+--------------+
```

> 📝 **채점 기준**: `emp_id = 'T999'` 조건은 반드시 충족해야 하며, 나머지 필드는 학생이 자유롭게 입력합니다.

연습 2-2 참고 정답

第 2 章 INSERT 练习题答案

练习 2-1 参考答案

题目：向 emp_test 表中插入自己的信息（员工编号 T999）。

```sql
USE unit5db;

-- 确认 emp_test 表已存在
SHOW TABLES LIKE 'emp_test';

-- 如不存在则先创建
CREATE TABLE IF NOT EXISTS emp_test (
    emp_id    STRING,
    emp_name  STRING,
    salary    DOUBLE,
    dept      STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 插入数据（示例，学生根据自己信息填写）
INSERT INTO TABLE emp_test
VALUES ('T999', 'ZhangSan', 12000, '大数据技术部');

-- 验证插入结果
SELECT * FROM emp_test WHERE emp_id = 'T999';
```

预期输出：

```
+--------+----------+--------+--------------+
| emp_id | emp_name | salary |     dept     |
+--------+----------+--------+--------------+
| T999   | ZhangSan | 12000  | 大数据技术部 |
+--------+----------+--------+--------------+
```

> 📝 **评分标准**：`emp_id = 'T999'` 必须满足，其他字段内容由学生自定义。

练习 2-2 参考答案

문제: `shanghai_employees` 테이블을 생성하고 상하이 직원 정보를 삽입한 후 인원수를 집계하세요.

题目：创建 shanghai_employees 表，插入上海员工信息并统计人数。

```sql
-- 단계 1: 테이블 생성
CREATE TABLE IF NOT EXISTS shanghai_employees (
    emp_id      STRING,
    emp_name    STRING,
    salary      DOUBLE,
    hire_date   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 단계 2: 상하이 직원 데이터 삽입
INSERT INTO TABLE shanghai_employees
SELECT emp_id, emp_name, salary, hire_date
FROM employees
WHERE city = '上海';

-- 단계 3: 결과 확인 및 인원수 집계
SELECT * FROM shanghai_employees;
SELECT COUNT(*) AS shanghai_emp_count FROM
shanghai_employees;
```

```sql
-- 步骤 1：创建表
CREATE TABLE IF NOT EXISTS shanghai_employees (
    emp_id      STRING,
    emp_name    STRING,
    salary      DOUBLE,
    hire_date   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 步骤 2：插入上海员工数据
INSERT INTO TABLE shanghai_employees
SELECT emp_id, emp_name, salary, hire_date
FROM employees
WHERE city = '上海';

-- 步骤 3：验证并统计
SELECT * FROM shanghai_employees;
SELECT COUNT(*) AS shanghai_emp_count FROM
shanghai_employees;
```

예상 출력:

```
+--------+----------+--------+------------+
| emp_id | emp_name | salary | hire_date  |
+--------+----------+--------+------------+
| E002   | 李娜      | 12000  | 2020-06-01 |
| E005   | 陈静      | 13000  | 2019-11-05 |
| E013   | 胡涛      | 12500  | 2020-10-12 |
+--------+----------+--------+------------+

+--------------------+
| shanghai_emp_count |
+--------------------+
| 3                  |
+--------------------+
```

预期输出：

```
+--------+----------+--------+------------+
| emp_id | emp_name | salary | hire_date  |
+--------+----------+--------+------------+
| E002   | 李娜      | 12000  | 2020-06-01 |
| E005   | 陈静      | 13000  | 2019-11-05 |
| E013   | 胡涛      | 12500  | 2020-10-12 |
+--------+----------+--------+------------+

+--------------------+
| shanghai_emp_count |
+--------------------+
| 3                  |
+--------------------+
```

연습 2-3 참고 정답

문제: `sales_summary` 테이블을 생성하고, 각 지역별 판매 데이터를 `INSERT OVERWRITE`로 저장하세요.

练习 2-3 参考答案

题目：创建 sales_summary 表，统计各区域销售数据并用 INSERT OVERWRITE 存入。

```sql
```

```sql
```

```
-- 단계 1: 통계 테이블 생성
CREATE TABLE IF NOT EXISTS sales_summary (
    region        STRING,
    total_amount  DOUBLE,
    avg_amount    DOUBLE,
    max_amount    DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 단계 2: 집계 후 덮어쓰기로 저장
INSERT OVERWRITE TABLE sales_summary
SELECT
    region,
    SUM(amount)             AS total_amount,
    ROUND(AVG(amount), 0)   AS avg_amount,
    MAX(amount)             AS max_amount
FROM sales
GROUP BY region
ORDER BY total_amount DESC;

-- 단계 3: 결과 확인
SELECT * FROM sales_summary;
```

예상 출력:

```
+--------+--------------+------------+------------+
| region | total_amount | avg_amount | max_amount |
+--------+--------------+------------+------------+
| 华南   | 854000       | 170800     | 200000     |
| 华北   | 619000       | 103167     | 135000     |
| 华东   | 676000       | 75111      | 110000     |
+--------+--------------+------------+------------+
```

연습 2-4 참고 정답 (도전 문제)

문제: FROM 다중 대상 삽입을 활용하여 `orders` 테이블의 완료 주문과 취소 주문을 각각 다른 테이블에 삽입하세요.

```sql
-- 단계 1: 대상 테이블 두 개 생성
CREATE TABLE IF NOT EXISTS completed_orders (
    order_id      STRING,
    customer_id   STRING,
    order_date    STRING,
```

```
-- 步骤 1：创建统计表
CREATE TABLE IF NOT EXISTS sales_summary (
    region        STRING,
    total_amount  DOUBLE,
    avg_amount    DOUBLE,
    max_amount    DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 步骤 2：统计并写入（使用 OVERWRITE 覆盖写入）
INSERT OVERWRITE TABLE sales_summary
SELECT
    region,
    SUM(amount)             AS total_amount,
    ROUND(AVG(amount), 0)   AS avg_amount,
    MAX(amount)             AS max_amount
FROM sales
GROUP BY region
ORDER BY total_amount DESC;

-- 步骤 3：验证结果
SELECT * FROM sales_summary;
```

预期输出：

```
+--------+--------------+------------+------------+
| region | total_amount | avg_amount | max_amount |
+--------+--------------+------------+------------+
| 华南   | 854000       | 170800     | 200000     |
| 华北   | 619000       | 103167     | 135000     |
| 华东   | 676000       | 75111      | 110000     |
+--------+--------------+------------+------------+
```

练习 2-4 参考答案（挑战题）

题目：利用 FROM 多目标插入，将 orders 表中"已完成"和"已取消"订单分别插入不同表。

```sql
-- 步骤 1：创建两张目标表
CREATE TABLE IF NOT EXISTS completed_orders (
    order_id      STRING,
    customer_id   STRING,
    order_date    STRING,
```

```
    total_amount  DOUBLE,
    emp_id        STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE IF NOT EXISTS cancelled_orders (
    order_id      STRING,
    customer_id   STRING,
    order_date    STRING,
    total_amount  DOUBLE,
    emp_id        STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 단계 2: FROM 다중 대상 삽입 (한 번 스캔으로 두 테이블에 쓰기)
FROM orders
INSERT INTO TABLE completed_orders
    SELECT order_id, customer_id, order_date, total_amount, emp_id
    WHERE status = '已完成'
INSERT INTO TABLE cancelled_orders
    SELECT order_id, customer_id, order_date, total_amount, emp_id
    WHERE status = '已取消';

-- 단계 3: 결과 확인
SELECT '완료 주문' AS type, COUNT(*) AS cnt FROM completed_orders
UNION ALL
SELECT '취소 주문', COUNT(*) FROM cancelled_orders;
```

예상 출력:

```
+-----------+-----+
|   type    | cnt |
+-----------+-----+
| 완료 주문 |  14 |
| 취소 주문 |   2 |
+-----------+-----+
```

제 3 장 SELECT 기본 조회 연습문제 정답

연습 3-1 참고 정답

```
    total_amount  DOUBLE,
    emp_id        STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE IF NOT EXISTS cancelled_orders (
    order_id      STRING,
    customer_id   STRING,
    order_date    STRING,
    total_amount  DOUBLE,
    emp_id        STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 步骤 2：使用 FROM 多目标插入（一次扫描，写入两张表）
FROM orders
INSERT INTO TABLE completed_orders
    SELECT order_id, customer_id, order_date, total_amount, emp_id
    WHERE status = '已完成'
INSERT INTO TABLE cancelled_orders
    SELECT order_id, customer_id, order_date, total_amount, emp_id
    WHERE status = '已取消';

-- 步骤 3：验证结果
SELECT '已完成订单' AS type, COUNT(*) AS cnt FROM completed_orders
UNION ALL
SELECT '已取消订单', COUNT(*) FROM cancelled_orders;
```

预期输出：

```
+------------+-----+
|    type    | cnt |
+------------+-----+
| 已完成订单 |  14 |
| 已取消订单 |   2 |
+------------+-----+
```

第 3 章 SELECT 基础查询 练习题答案

练习 3-1 参考答案

문제: 30 세 이상 남성 직원을 조회하고, 월급 내림차순으로 정렬하세요.

```sql
SELECT
    emp_name AS 이름,
    age      AS 나이,
    city     AS 소재도시,
    salary   AS 월급
FROM employees
WHERE age >= 30
  AND gender = '男'
ORDER BY salary DESC;
```

예상 출력:

```
+--------+------+------+-------+
|  이름  | 나이 | 소재도시 | 월급  |
+--------+------+------+-------+
| 朱峰   |  41  | 广州  | 20000 |
| 王强   |  35  | 北京  | 18000 |
| 吴刚   |  38  | 杭州  | 17000 |
| 杨帆   |  33  | 广州  | 14000 |
| 张伟   |  32  | 北京  | 15000 |
+--------+------+------+-------+
```

연습 3-2 참고 정답

문제: 2023 년 2 월의 모든 판매 기록을 조회하고, 금액 내림차순으로 정렬하세요.

```sql
SELECT
    sale_id      AS 판매번호,
    emp_id       AS 직원번호,
    amount       AS 판매금액,
    product_type AS 제품유형
FROM sales
WHERE sale_date LIKE '2023-02%'
ORDER BY amount DESC;
```

예상 출력:

```
+----------+----------+----------+----------+
```

题目：查询 30 岁以上男性员工，按月薪降序排列。

```sql
SELECT
    emp_name AS 姓名,
    age      AS 年龄,
    city     AS 所在城市,
    salary   AS 月薪
FROM employees
WHERE age >= 30
  AND gender = '男'
ORDER BY salary DESC;
```

预期输出：

```
+--------+------+------+-------+
|  姓名  | 年龄 | 城市 | 月薪  |
+--------+------+------+-------+
| 朱峰   | 41   | 广州 | 20000 |
| 吴刚   | 38   | 杭州 | 17000 |
| 王强   | 35   | 北京 | 18000 |
| 杨帆   | 33   | 广州 | 14000 |
| 张伟   | 32   | 北京 | 15000 |
+--------+------+------+-------+
```

练习 3-2 参考答案

题目：查询 2023 年 2 月份的所有销售记录，按金额降序排列。

```sql
SELECT
    sale_id      AS 销售编号,
    emp_id       AS 员工编号,
    amount       AS 销售金额,
    product_type AS 产品类型
FROM sales
WHERE sale_date LIKE '2023-02%'
ORDER BY amount DESC;
```

预期输出：

```
+---------+--------+---------+---------+
```

```
| 판매번호 | 직원번호 | 판매금액 | 제품유형 |
+----------+----------+----------+----------+
| S017     | E015     | 200000   | 服务项目 |
| S018     | E006     | 168000   | 硬件设备 |
| S013     | E003     | 135000   | 软件产品 |
| S004     | E002     | 91000    | 软件产品 |
| S005     | E005     | 75000    | 服务项目 |
| S012     | E014     | 78000    | 软件产品 |
| S006     | E013     | 55000    | 硬件设备 |
+----------+----------+----------+----------+
```

연습 3-3 참고 정답

문제: 완료（已完成）상태이면서 총금액이 100,000 원을 초과하는 주문을 조회하세요.

```sql
SELECT
    order_id     AS 주문번호,
    customer_id  AS 고객번호,
    order_date   AS 주문날짜,
    total_amount AS 총금액
FROM orders
WHERE status = '已完成'
  AND total_amount > 100000
ORDER BY total_amount DESC;
```

예상 출력:

```
+----------+----------+------------+----------+
| 주문번호 | 고객번호 |  주문날짜  |  총금액  |
+----------+----------+------------+----------+
| O1017    | C012     | 2023-02-08 | 200000   |
| O1019    | C013     | 2023-03-12 | 185000   |
| O1018    | C011     | 2023-02-25 | 168000   |
| O1020    | C014     | 2023-03-28 | 156000   |
| O1016    | C011     | 2023-01-15 | 145000   |
| O1013    | C007     | 2023-02-20 | 135000   |
| O1007    | C002     | 2023-03-10 | 110000   |
| O1014    | C010     | 2023-03-05 | 102000   |
+----------+----------+------------+----------+
```

연습 3-4 참고 정답

```
| 销售编号 | 员工编号 | 销售金额 | 产品类型 |
+----------+----------+----------+----------+
| S017     | E015     | 200000   | 服务项目 |
| S018     | E006     | 168000   | 硬件设备 |
| S013     | E003     | 135000   | 软件产品 |
| S004     | E002     | 91000    | 软件产品 |
| S005     | E005     | 75000    | 服务项目 |
| S012     | E014     | 78000    | 软件产品 |
| S006     | E013     | 55000    | 硬件设备 |
+----------+----------+----------+----------+
```

练习 3-3 参考答案

题目：查询"已完成"且总金额超过 100000 元的订单。

```sql
SELECT
    order_id     AS 订单编号,
    customer_id  AS 客户编号,
    order_date   AS 订单日期,
    total_amount AS 总金额
FROM orders
WHERE status = '已完成'
  AND total_amount > 100000
ORDER BY total_amount DESC;
```

预期输出：

```
+--------+--------+------------+---------+
|订单编号|客户编号|  订单日期  |  总金额 |
+--------+--------+------------+---------+
| O1017  | C012   | 2023-02-08 | 200000  |
| O1019  | C013   | 2023-03-12 | 185000  |
| O1018  | C011   | 2023-02-25 | 168000  |
| O1016  | C011   | 2023-01-15 | 145000  |
| O1013  | C007   | 2023-02-20 | 135000  |
| O1020  | C014   | 2023-03-28 | 156000  |
| O1007  | C002   | 2023-03-10 | 110000  |
| O1014  | C010   | 2023-03-05 | 102000  |
+--------+--------+------------+---------+
```

练习 3-4 参考答案

문제: CASE WHEN 을 사용하여 직원 나이를 구간별로 분류하세요.

```sql
SELECT
    emp_name AS 이름,
    age      AS 나이,
    CASE
        WHEN age < 25  THEN '신세대'
        WHEN age <= 35 THEN '핵심인력'
        ELSE                '시니어직원'
    END AS 연령구간
FROM employees
ORDER BY age ASC;
```

예상 출력:

```
+--------+------+------------+
|  이름  | 나이 |  연령구간  |
+--------+------+------------+
| 孙浩   |  24  | 신세대     |
| 刘洋   |  26  | 핵심인력   |
| 周梅   |  27  | 핵심인력   |
| 李娜   |  28  | 핵심인력   |
| 胡涛   |  28  | 핵심인력   |
| 赵磊   |  29  | 핵심인력   |
| 林晓   |  29  | 핵심인력   |
| 陈静   |  30  | 핵심인력   |
| 郑芳   |  31  | 핵심인력   |
| 张伟   |  32  | 핵심인력   |
| 杨帆   |  33  | 핵심인력   |
| 王强   |  35  | 핵심인력   |
| 马丽   |  36  | 시니어직원 |
| 吴刚   |  38  | 시니어직원 |
| 朱峰   |  41  | 시니어직원 |
+--------+------+------------+
```

제 4 장 집계 조회 연습문제 정답

题目：用 CASE WHEN 对员工年龄分段。

```sql
SELECT
    emp_name AS 姓名,
    age      AS 年龄,
    CASE
        WHEN age < 25  THEN '新生代'
        WHEN age <= 35 THEN '中坚力量'
        ELSE                '资深员工'
    END AS 年龄段
FROM employees
ORDER BY age ASC;
```

预期输出：

```
+--------+------+----------+
|  姓名  | 年龄 |  年龄段  |
+--------+------+----------+
| 孙浩   |  24  | 新生代   |
| 刘洋   |  26  | 中坚力量 |
| 周梅   |  27  | 中坚力量 |
| 李娜   |  28  | 中坚力量 |
| 胡涛   |  28  | 中坚力量 |
| 赵磊   |  29  | 中坚力量 |
| 林晓   |  29  | 中坚力量 |
| 陈静   |  30  | 中坚力量 |
| 郑芳   |  31  | 中坚力量 |
| 张伟   |  32  | 中坚力量 |
| 杨帆   |  33  | 中坚力量 |
| 王强   |  35  | 中坚力量 |
| 马丽   |  36  | 资深员工 |
| 吴刚   |  38  | 资深员工 |
| 朱峰   |  41  | 资深员工 |
+--------+------+----------+
```

第 4 章 聚合查询练习题答案

연습 4-1 참고 정답

문제: 부서별·성별로 직원 수와 평균 월급을 집계하세요.

```sql
SELECT
    dept_id             AS 부서번호,
    gender              AS 성별,
    COUNT(*)            AS 직원수,
    ROUND(AVG(salary),0) AS 평균월급
FROM employees
GROUP BY dept_id, gender
ORDER BY dept_id ASC, gender ASC;
```

예상 출력:

```
+----------+------+--------+----------+
| 부서번호 | 성별 | 직원수 | 평균월급 |
+----------+------+--------+----------+
| D001     | 女   |      1 |    14500 |
| D001     | 男   |      3 |    16333 |
| D002     | 女   |      2 |    12500 |
| D002     | 男   |      1 |    12500 |
| D003     | 女   |      2 |    10000 |
| D004     | 男   |      2 |    17000 |
| D005     | 女   |      1 |    10000 |
| D005     | 男   |      1 |     8500 |
| D006     | 女   |      1 |    16500 |
| D006     | 男   |      1 |    17000 |
+----------+------+--------+----------+
```

연습 4-2 참고 정답

문제: 각 직원의 총 판매액을 집계하고, 총 판매액이 200,000 원을 초과하는 직원만 표시하세요.

```sql
SELECT
    emp_id          AS 직원번호,
    COUNT(*)        AS 판매횟수,
    SUM(amount)     AS 총판매액
FROM sales
GROUP BY emp_id
```

练习 4-1 参考答案

题目：按部门和性别统计员工人数和平均薪资。

```sql
SELECT
    dept_id             AS 部门编号,
    gender              AS 性别,
    COUNT(*)            AS 员工人数,
    ROUND(AVG(salary),0) AS 平均薪资
FROM employees
GROUP BY dept_id, gender
ORDER BY dept_id ASC, gender ASC;
```

预期输出：

```
+--------+------+----------+----------+
|部门编号| 性别 | 员工人数 | 平均薪资 |
+--------+------+----------+----------+
| D001   | 女   | 1        | 14500    |
| D001   | 男   | 3        | 16333    |
| D002   | 女   | 2        | 12500    |
| D002   | 男   | 1        | 12500    |
| D003   | 女   | 2        | 10000    |
| D004   | 男   | 2        | 17000    |
| D005   | 女   | 1        | 10000    |
| D005   | 男   | 1        | 8500     |
| D006   | 女   | 1        | 16500    |
| D006   | 男   | 1        | 17000    |
+--------+------+----------+----------+
```

练习 4-2 参考答案

题目：统计各员工总销售额，只显示超过 200000 元的员工。

```sql
SELECT
    emp_id          AS 员工编号,
    COUNT(*)        AS 销售次数,
    SUM(amount)     AS 总销售额
FROM sales
GROUP BY emp_id
```

```
HAVING SUM(amount) > 200000
ORDER BY 총판매액 DESC;
```

예상 출력:

```
+----------+----------+----------+
| 직원번호 | 판매횟수 | 총판매액 |
+----------+----------+----------+
| E006     |        3 |   469000 |
| E015     |        2 |   385000 |
| E003     |        2 |   255000 |
+----------+----------+----------+
```

연습 4-3 참고 정답

문제: `orders` 테이블에서 각 상태별 주문 수와 총금액을 집계하고, 주문 수가 5 건을 초과하는 상태만 표시하세요.

```sql
SELECT
    status           AS 주문상태,
    COUNT(*)         AS 주문수,
    SUM(total_amount) AS 총금액
FROM orders
GROUP BY status
HAVING COUNT(*) > 5
ORDER BY 주문수 DESC;
```

예상 출력:

```
+----------+--------+----------+
| 주문상태 | 주문수 |  총금액  |
+----------+--------+----------+
| 已完成   |     14 |  1605000 |
+----------+--------+----------+
```

> 📝 **해설**: 취소（已取消）는 2 건, 처리 중（处理中）은 3 건으로 모두 > 5 조건을 충족하지 못합니다. 완료（已完成）만 14 건으로 조건을 만족합니다.

```
HAVING SUM(amount) > 200000
ORDER BY 总销售额 DESC;
```

预期输出：

```
+--------+----------+----------+
|员工编号| 销售次数 | 总销售额 |
+--------+----------+----------+
| E015   | 2        | 385000   |
| E006   | 3        | 469000   |
| E003   | 2        | 255000   |
+--------+----------+----------+
```

练习 4-3 参考答案

题目：统计 orders 各状态的订单数和总金额，只保留订单数>5 的状态。

```sql
SELECT
    status       AS 订单状态,
    COUNT(*)     AS 订单数量,
    SUM(total_amount) AS 总金额
FROM orders
GROUP BY status
HAVING COUNT(*) > 5
ORDER BY 订单数量 DESC;
```

预期输出：

```
+----------+----------+----------+
| 订单状态 | 订单数量 |  总金额  |
+----------+----------+----------+
| 已完成   | 14       | 1605000  |
+----------+----------+----------+
```

> 📝 **解析**："已取消"有 2 条，"处理中"有 3 条，都不满足 > 5 的条件；只有"已完成"有 14 条，满足条件。

연습 4-4 참고 정답 (도전 문제)

문제: 직원 수가 2 명을 초과하는 도시만 평균 월급 내림차순으로 집계하세요.

```sql
SELECT
    city                AS 도시,
    COUNT(*)            AS 직원수,
    ROUND(AVG(salary),0) AS 평균월급,
    MAX(salary)         AS 최고월급
FROM employees
GROUP BY city
HAVING COUNT(*) > 2
ORDER BY 평균월급 DESC;
```

예상 출력:

```
+------+--------+----------+----------+
| 도시 | 직원수 | 평균월급 | 최고월급 |
+------+--------+----------+----------+
| 北京 |      6 |    14000 |    18000 |
| 上海 |      3 |    12500 |    13000 |
+------+--------+----------+----------+
```

제 5 장 JOIN 연습문제 정답

练习 4-4 参考答案（挑战题）

题目：统计各城市员工人数>2、按平均薪资降序排列。

```sql
SELECT
    city                AS 城市,
    COUNT(*)            AS 员工人数,
    ROUND(AVG(salary),0) AS 平均薪资,
    MAX(salary)         AS 最高薪资
FROM employees
GROUP BY city
HAVING COUNT(*) > 2
ORDER BY 平均薪资 DESC;
```

预期输出：

```
+------+----------+----------+----------+
| 城市 | 员工人数 | 平均薪资 | 最高薪资 |
+------+----------+----------+----------+
| 北京 | 6        | 14000    | 18000    |
| 上海 | 3        | 12500    | 13000    |
+------+----------+----------+----------+
```

实际预期（满足>2 人的城市）：

```
+------+----------+----------+----------+
| 城市 | 员工人数 | 平均薪资 | 最高薪资 |
+------+----------+----------+----------+
| 北京 | 6        | 14000    | 18000    |
| 上海 | 3        | 12500    | 13000    |
+------+----------+----------+----------+
```

第 5 章 JOIN 练习题答案

연습 5-1 참고 정답

문제: INNER JOIN 으로 직원과 부서를 연결하여 이름·부서명·도시·월급을 표시하고, 월급 내림차순 상위 10 명을 조회하세요.

```sql
SELECT
    e.emp_name   AS 직원이름,
    d.dept_name  AS 부서명,
    e.city       AS 소재도시,
    e.salary     AS 월급
FROM employees e
INNER JOIN departments d ON e.dept_id = d.dept_id
ORDER BY e.salary DESC
LIMIT 10;
```

예상 출력:

```
+----------+-----------+----------+-------+
| 직원이름 |   부서명  | 소재도시 | 월급  |
+----------+-----------+----------+-------+
| 朱峰     | 财务管理部 | 广州     | 20000 |
| 王强     | 技术研发部 | 北京     | 18000 |
| 吴刚     | 产品设计部 | 杭州     | 17000 |
| 马丽     | 产品设计部 | 杭州     | 16500 |
| 赵磊     | 技术研发部 | 北京     | 16000 |
| 张伟     | 技术研发部 | 北京     | 15000 |
| 林晓     | 技术研发部 | 北京     | 14500 |
| 杨帆     | 财务管理部 | 广州     | 14000 |
| 陈静     | 市场营销部 | 上海     | 13000 |
| 胡涛     | 市场营销部 | 上海     | 12500 |
+----------+-----------+----------+-------+
```

练习 5-1 参考答案

题目：INNER JOIN 员工与部门，显示姓名、部门名称、城市、月薪，前 10 名按薪资降序。

```sql
SELECT
    e.emp_name   AS 员工姓名,
    d.dept_name  AS 部门名称,
    e.city       AS 所在城市,
    e.salary     AS 月薪
FROM employees e
INNER JOIN departments d ON e.dept_id = d.dept_id
ORDER BY e.salary DESC
LIMIT 10;
```

预期输出：

```
+--------+----------+------+-------+
| 员工姓名 | 部门名称 | 城市 | 月薪  |
+--------+----------+------+-------+
| 朱峰   | 财务管理部 | 广州 | 20000 |
| 王强   | 技术研发部 | 北京 | 18000 |
| 吴刚   | 产品设计部 | 杭州 | 17000 |
| 马丽   | 产品设计部 | 杭州 | 16500 |
| 赵磊   | 技术研发部 | 北京 | 16000 |
| 张伟   | 技术研发部 | 北京 | 15000 |
| 林晓   | 技术研发部 | 北京 | 14500 |
| 杨帆   | 财务管理部 | 广州 | 14000 |
| 陈静   | 市场营销部 | 上海 | 13000 |
| 胡涛   | 市场营销部 | 上海 | 12500 |
+--------+----------+------+-------+
```

연습 5-2 참고 정답

문제: LEFT JOIN 으로 모든 직원과 판매 기록을 연결하여 직원별 판매 횟수를 집계하세요.

```sql
SELECT
    e.emp_id                    AS 직원번호,
```

练习 5-2 参考答案

题目：LEFT JOIN 员工与销售记录，统计每个员工的销售次数。

```sql
SELECT
    e.emp_id     AS 员工编号,
```

```
    e.emp_name                  AS 직원이름,
    COUNT(s.sale_id)            AS 판매횟수,
    COALESCE(SUM(s.amount), 0)  AS 총판매액
FROM employees e
LEFT JOIN sales s ON e.emp_id = s.emp_id
GROUP BY e.emp_id, e.emp_name
ORDER BY 판매횟수 DESC;
```

예상 출력 (일부):

```
+----------+----------+----------+----------+
| 직원번호 | 직원이름 | 판매횟수 | 총판매액 |
+----------+----------+----------+----------+
| E006     | 杨帆     |        3 |   469000 |
| E002     | 李娜     |        3 |   286000 |
| E005     | 陈静     |        3 |   225000 |
| E013     | 胡涛     |        3 |   165000 |
| E015     | 朱峰     |        2 |   385000 |
| E003     | 王强     |        2 |   255000 |
| E007     | 赵磊     |        2 |   197000 |
| E014     | 林晓     |        2 |   167000 |
| E001     | 张伟     |        0 |        0 |
... (판매 기록 없는 직원은 판매 횟수 0)
+----------+----------+----------+----------+
```

> 💡 **핵심 포인트**: `COALESCE(SUM(s.amount), 0)`으로 NULL 을 0 으로 변환합니다. `COUNT(*)`이 아닌 `COUNT(s.sale_id)`를 사용해야 LEFT JOIN 에서 NULL 행이 카운트되지 않습니다.

연습 5-3 참고 정답

문제: 3 개 테이블을 조인하여 각 부서별 제품 유형별 총 판매액을 집계하세요.

```sql
SELECT
    d.dept_name     AS 부서명,
    s.product_type AS 제품유형,
    COUNT(s.sale_id)  AS 판매횟수,
    SUM(s.amount)     AS 총판매액
FROM departments d
INNER JOIN employees e ON d.dept_id = e.dept_id
```

```
    e.emp_name    AS 员工姓名,
    COUNT(s.sale_id)          AS 销售次数,
    COALESCE(SUM(s.amount), 0) AS 销售总额
FROM employees e
LEFT JOIN sales s ON e.emp_id = s.emp_id
GROUP BY e.emp_id, e.emp_name
ORDER BY 销售次数 DESC;
```

预期输出（部分）：

```
+--------+--------+----------+----------+
|员工编号| 员工姓名 | 销售次数 | 销售总额 |
+--------+--------+----------+----------+
| E006   | 杨帆   | 3        | 469000   |
| E002   | 李娜   | 3        | 286000   |
| E005   | 陈静   | 3        | 225000   |
| E013   | 胡涛   | 3        | 165000   |
| E015   | 朱峰   | 2        | 385000   |
| E003   | 王强   | 2        | 255000   |
| E007   | 赵磊   | 2        | 197000   |
| E014   | 林晓   | 2        | 167000   |
| E001   | 张伟   | 0        | 0        |
... (无销售记录的员工销售次数为 0)
+--------+--------+----------+----------+
```

> 💡 **要点**：使用 `COALESCE(SUM(s.amount), 0)` 将 NULL 转为 0；`COUNT(s.sale_id)` 而非 `COUNT(*)`，避免左连接中 NULL 行被计数。

练习 5-3 参考答案

题目：连接三张表，统计每个部门下各产品类型的销售总额。

```sql
SELECT
    d.dept_name    AS 部门名称,
    s.product_type AS 产品类型,
    COUNT(s.sale_id)  AS 销售次数,
    SUM(s.amount)     AS 总销售额
FROM departments d
INNER JOIN employees e ON d.dept_id = e.dept_id
```

```
INNER JOIN sales s     ON e.emp_id = s.emp_id
GROUP BY d.dept_name, s.product_type
ORDER BY d.dept_name, 총판매액 DESC;
```

예상 출력 (일부):

```
+------------+----------+----------+----------+
|   부서명   | 제품유형 | 판매횟수 | 총판매액 |
+------------+----------+----------+----------+
| 财务管理部 | 软件产品 |        2 |   345000 |
| 财务管理部 | 硬件设备 |        2 |   313000 |
| 财务管理部 | 服务项目 |        1 |   200000 |
| 技术研发部 | 软件产品 |        4 |   424500 |
| 技术研发部 | 硬件设备 |        2 |   173000 |
| 技术研发部 | 服务项目 |        1 |   102000 |
| 市场营销部 | 软件产品 |        4 |   349000 |
...
+------------+----------+----------+----------+
```

연습 5-4 참고 정답 (도전 문제)

문제: 각 고객의 주문 총금액·주문 건수, 그리고 가장 많이 담당한 직원 이름을 조회하세요.

```sql
-- 방안: 고객별 가장 많이 나타나는 직원을 집계하여 연결
SELECT
    o.customer_id          AS 고객번호,
    COUNT(o.order_id)      AS 주문건수,
    SUM(o.total_amount)    AS 주문총금액,
    e.emp_name             AS 주담당직원
FROM orders o
INNER JOIN employees e ON o.emp_id = e.emp_id
GROUP BY o.customer_id, o.emp_id, e.emp_name
ORDER BY o.customer_id, COUNT(o.order_id) DESC;
```

> 📝 **설명**: Hive 의 분석 함수로 "가장 많이 나타나는 직원"을 더 정확하게 처리할 수 있지만, 위 쿼리는 실용적인 근사 해법입니다. 고급 방법은 윈도우 함수를 사용합니다 (종합 문제 정답 참고).


```
INNER JOIN sales s     ON e.emp_id = s.emp_id
GROUP BY d.dept_name, s.product_type
ORDER BY d.dept_name, 总销售额 DESC;
```


预期输出（部分）：

```
+----------+----------+----------+----------+
| 部门名称 | 产品类型 | 销售次数 | 总销售额 |
+----------+----------+----------+----------+
| 财务管理部 | 软件产品 | 2        | 345000   |
| 财务管理部 | 服务项目 | 1        | 200000   |
| 财务管理部 | 硬件设备 | 2        | 313000   |
| 技术研发部 | 软件产品 | 4        | 424500   |
| 技术研发部 | 服务项目 | 1        | 102000   |
| 技术研发部 | 硬件设备 | 2        | 173000   |
| 市场营销部 | 软件产品 | 4        | 349000   |
...
+----------+----------+----------+----------+
```

练习 5-4 参考答案（挑战题）

题目：查询每个客户的订单总金额、订单数量及负责订单最多的员工姓名。

```sql
-- 方案：先统计每个客户最常用的员工，再关联
SELECT
    o.customer_id          AS 客户编号,
    COUNT(o.order_id)      AS 订单数量,
    SUM(o.total_amount)    AS 订单总金额,
    e.emp_name             AS 主要负责员工
FROM orders o
INNER JOIN employees e ON o.emp_id = e.emp_id
GROUP BY o.customer_id, o.emp_id, e.emp_name
ORDER BY o.customer_id, COUNT(o.order_id) DESC;
```

> 📝 **说明**：Hive 的分析函数可以更精确处理"最常出现的员工"，但以上查询是一个实用的近似解法。对于高级版本，可使用窗口函数（见综合题解答）。

제 6 장 서브쿼리 연습문제 정답

연습 6-1 참고 정답

문제: 월급이 가장 높은 직원이 속한 부서의 모든 직원 정보를 조회하세요.

```sql
SELECT
    emp_id   AS 직원번호,
    emp_name AS 직원이름,
    salary   AS 월급,
    dept_id  AS 부서번호
FROM employees
WHERE dept_id = (
    -- 서브쿼리: 월급이 가장 높은 직원의 부서 번호 조회
    SELECT dept_id
    FROM employees
    WHERE salary = (SELECT MAX(salary) FROM employees)
)
ORDER BY salary DESC;
```

예상 출력 (월급 최고는 朱峰 20,000 원, 부서 D004 财务管理部):

```
+----------+----------+-------+----------+
| 직원번호 | 직원이름 | 월급  | 부서번호 |
+----------+----------+-------+----------+
| E015     | 朱峰     | 20000 | D004     |
| E006     | 杨帆     | 14000 | D004     |
+----------+----------+-------+----------+
```

연습 6-2 참고 정답

문제: 주문 총금액이 모든 주문의 평균 총금액보다 높은 주문을 조회하세요.

第 6 章 子查询练习题答案

练习 6-1 参考答案

题目：找出薪资最高的员工所在部门的所有员工。

```sql
SELECT
    emp_id   AS 员工编号,
    emp_name AS 员工姓名,
    salary   AS 月薪,
    dept_id  AS 部门编号
FROM employees
WHERE dept_id = (
    -- 子查询：找出薪资最高的员工所在部门
    SELECT dept_id
    FROM employees
    WHERE salary = (SELECT MAX(salary) FROM employees)
)
ORDER BY salary DESC;
```

预期输出（薪资最高的是朱峰，20000，部门 D004 财务管理部）：

```
+--------+--------+-------+--------+
|员工编号| 员工姓名 | 月薪  |部门编号|
+--------+--------+-------+--------+
| E015   | 朱峰   | 20000 | D004   |
| E006   | 杨帆   | 14000 | D004   |
+--------+--------+-------+--------+
```

练习 6-2 参考答案

题目：查询订单总金额高于平均总金额的订单。

```sql
SELECT
    order_id      AS 주문번호,
    customer_id   AS 고객번호,
    total_amount  AS 총금액,
    status        AS 상태
FROM orders
WHERE total_amount > (SELECT AVG(total_amount)
FROM orders)
ORDER BY total_amount DESC;
```

평균 총금액 확인:

```sql
-- 평균값 먼저 확인
SELECT ROUND(AVG(total_amount), 0) AS avg_amount
FROM orders;
-- 예상 약 109,850 원
```

예상 출력 (평균 초과 주문) :

```
+----------+----------+----------+--------+
| 주문번호 | 고객번호 |  총금액  |  상태  |
+----------+----------+----------+--------+
| O1017    | C012     | 200000   | 已完成 |
| O1019    | C013     | 185000   | 已完成 |
| O1018    | C011     | 168000   | 已完成 |
| O1020    | C014     | 156000   | 已完成 |
| O1016    | C011     | 145000   | 已完成 |
| O1013    | C007     | 135000   | 已完成 |
| O1010    | C007     | 120000   | 已完成 |
| O1007    | C002     | 110000   | 已完成 |
+----------+----------+----------+--------+
```

연습 6-3 참고 정답

문제: NOT IN 서브쿼리를 사용하여 주문을 처리한 적 없는 직원을 조회하세요.

```sql
SELECT
    emp_id   AS 직원번호,
    emp_name AS 직원이름,
    dept_id  AS 부서번호,
    salary   AS 월급
```

```sql
SELECT
    order_id      AS 订单编号,
    customer_id   AS 客户编号,
    total_amount  AS 总金额,
    status        AS 状态
FROM orders
WHERE total_amount > (SELECT AVG(total_amount)
FROM orders)
ORDER BY total_amount DESC;
```

平均总金额计算： 所有 20 条订单总金额之和 / 20

```sql
-- 可先查看平均值
SELECT ROUND(AVG(total_amount), 0) AS avg_amount
FROM orders;
-- 预期约 109850 元
```

预期输出（高于均值的订单）：

```
+--------+--------+----------+--------+
|订单编号|客户编号|  总金额  |  状态  |
+--------+--------+----------+--------+
| O1017  | C012   | 200000   | 已完成 |
| O1019  | C013   | 185000   | 已完成 |
| O1018  | C011   | 168000   | 已完成 |
| O1020  | C014   | 156000   | 已完成 |
| O1016  | C011   | 145000   | 已完成 |
| O1013  | C007   | 135000   | 已完成 |
| O1010  | C007   | 120000   | 已完成 |
| O1007  | C002   | 110000   | 已完成 |
+--------+--------+----------+--------+
```

练习 6-3 参考答案

题目：找出没有接过订单的员工（NOT IN 子查询）。

```sql
SELECT
    emp_id   AS 员工编号,
    emp_name AS 员工姓名,
    dept_id  AS 部门编号,
    salary   AS 月薪
```

```
FROM employees
WHERE emp_id NOT IN (
    SELECT DISTINCT emp_id FROM orders
)
ORDER BY dept_id, emp_id;
```

예상 출력:

```
+----------+----------+----------+-------+
| 직원번호 | 직원이름 | 부서번호 | 월급  |
+----------+----------+----------+-------+
| E001     | 张伟     | D001     | 15000 |
| E004     | 刘洋     | D003     |  9000 |
| E008     | 周梅     | D005     | 10000 |
| E009     | 吴刚     | D006     | 17000 |
| E010     | 郑芳     | D003     | 11000 |
| E011     | 孙浩     | D005     |  8500 |
| E012     | 马丽     | D006     | 16500 |
+----------+----------+----------+-------+
```

제 7 장 데이터 내보내기 연습문제 정답

연습 7-1 참고 정답

문제: 완료（已完成）주문을 로컬 디렉터리로 내보내세요.

```sql
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/completed_orders'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT
    order_id,
    customer_id,
    order_date,
    total_amount,
    status,
    emp_id
FROM orders
WHERE status = '已完成'
ORDER BY total_amount DESC;
```

검증（터미널에서 실행）:

```
FROM employees
WHERE emp_id NOT IN (
    SELECT DISTINCT emp_id FROM orders
)
ORDER BY dept_id, emp_id;
```

预期输出（orders 中只出现了部分员工的编号）：

```
+--------+--------+--------+-------+
|员工编号| 员工姓名 |部门编号| 月薪 |
+--------+--------+--------+-------+
| E001   | 张伟   | D001   | 15000 |
| E004   | 刘洋   | D003   | 9000  |
| E008   | 周梅   | D005   | 10000 |
| E009   | 吴刚   | D006   | 17000 |
| E010   | 郑芳   | D003   | 11000 |
| E011   | 孙浩   | D005   | 8500  |
| E012   | 马丽   | D006   | 16500 |
+--------+--------+--------+-------+
```

第 7 章 数据导出练习题答案

练习 7-1 参考答案

题目：将"已完成"的订单导出到本地目录。

```sql
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/completed_orders'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT
    order_id,
    customer_id,
    order_date,
    total_amount,
    status,
    emp_id
FROM orders
WHERE status = '已完成'
ORDER BY total_amount DESC;
```

验证（在终端执行）：

```bash
ls -la /home/hadoop/output/unit5/completed_orders/
cat /home/hadoop/output/unit5/completed_orders/*
```

예상 출력 (14 행 데이터):

```
O1017,C012,2023-02-08,200000.0,已完成,E015
O1019,C013,2023-03-12,185000.0,已完成,E015
O1018,C011,2023-02-25,168000.0,已完成,E006
...
```

연습 7-2 참고 정답

문제: 부서별 직원 요약 통계를 HDFS 로 내보내세요.

```sql
INSERT OVERWRITE DIRECTORY
'/user/hive/output/unit5/dept_summary'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT
    d.dept_name                AS dept_name,
    COUNT(e.emp_id)            AS emp_count,
    COALESCE(SUM(e.salary), 0) AS total_salary
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_name
ORDER BY total_salary DESC;
```

검증 (터미널에서 실행):

```bash
hdfs dfs -ls /user/hive/output/unit5/dept_summary
hdfs dfs -cat
/user/hive/output/unit5/dept_summary/*
```

제 8 장 종합 연습문제 정답

종합 문제 1 참고 정답: 직원 성과 분석 보고서

```sql
USE unit5db;
```


```bash
ls -la /home/hadoop/output/unit5/completed_orders/
cat /home/hadoop/output/unit5/completed_orders/*
```

预期输出（应有 14 行数据）：

```
O1017,C012,2023-02-08,200000.0,已完成,E015
O1019,C013,2023-03-12,185000.0,已完成,E015
O1018,C011,2023-02-25,168000.0,已完成,E006
...
```

练习 7-2 参考答案

题目：将各部门员工汇总统计导出到 HDFS。

```sql
INSERT OVERWRITE DIRECTORY
'/user/hive/output/unit5/dept_summary'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT
    d.dept_name                AS dept_name,
    COUNT(e.emp_id)            AS emp_count,
    COALESCE(SUM(e.salary), 0) AS total_salary
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_name
ORDER BY total_salary DESC;
```

验证（在终端执行）：

```bash
hdfs dfs -ls /user/hive/output/unit5/dept_summary
hdfs dfs -cat
/user/hive/output/unit5/dept_summary/*
```

第 8 章 综合练习题答案

综合题 1 参考答案：员工绩效分析报表

```sql
USE unit5db;
```

```
-- 단계 1: 종합 테이블 생성
CREATE TABLE IF NOT EXISTS emp_performance_report
(
    emp_id        STRING  COMMENT '직원 번호',
    emp_name      STRING  COMMENT '직원 이름',
    dept_name     STRING  COMMENT '부서명',
    salary        DOUBLE  COMMENT '월급',
    sale_count    INT     COMMENT '판매 횟수',
    total_sales   DOUBLE  COMMENT '총 판매액',
    order_count   INT     COMMENT '담당 주문 수',
    salary_level  STRING  COMMENT '월급 등급'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 단계 2: 데이터 채우기
INSERT OVERWRITE TABLE emp_performance_report
SELECT
    e.emp_id,
    e.emp_name,
    d.dept_name,
    e.salary,
    COALESCE(s.sale_count,  0) AS sale_count,
    COALESCE(s.total_sales, 0) AS total_sales,
    COALESCE(o.order_count, 0) AS order_count,
    CASE
        WHEN e.salary >= 16000 THEN '고'
        WHEN e.salary >= 12000 THEN '중'
        ELSE                        '저'
    END AS salary_level
FROM employees e
INNER JOIN departments d ON e.dept_id = d.dept_id
-- 판매 집계 LEFT JOIN (서브쿼리)
LEFT JOIN (
    SELECT emp_id, COUNT(*) AS sale_count,
SUM(amount) AS total_sales
    FROM sales
    GROUP BY emp_id
) s ON e.emp_id = s.emp_id
-- 주문 집계 LEFT JOIN (서브쿼리)
LEFT JOIN (
    SELECT emp_id, COUNT(*) AS order_count
    FROM orders
    GROUP BY emp_id
) o ON e.emp_id = o.emp_id
ORDER BY total_sales DESC;

-- 단계 3: 결과 확인
SELECT * FROM emp_performance_report;
```

```
-- 步骤 1：创建汇总表
CREATE TABLE IF NOT EXISTS emp_performance_report
(
    emp_id        STRING  COMMENT '员工编号',
    emp_name      STRING  COMMENT '员工姓名',
    dept_name     STRING  COMMENT '部门名称',
    salary        DOUBLE  COMMENT '月薪',
    sale_count    INT     COMMENT '销售次数',
    total_sales   DOUBLE  COMMENT '总销售额',
    order_count   INT     COMMENT '负责订单数',
    salary_level  STRING  COMMENT '薪资等级'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 步骤 2：填充数据
INSERT OVERWRITE TABLE emp_performance_report
SELECT
    e.emp_id,
    e.emp_name,
    d.dept_name,
    e.salary,
    COALESCE(s.sale_count,  0) AS sale_count,
    COALESCE(s.total_sales, 0) AS total_sales,
    COALESCE(o.order_count, 0) AS order_count,
    CASE
        WHEN e.salary >= 16000 THEN '高'
        WHEN e.salary >= 12000 THEN '中'
        ELSE                        '低'
    END AS salary_level
FROM employees e
INNER JOIN departments d ON e.dept_id = d.dept_id
-- 左连接销售汇总（子查询）
LEFT JOIN (
    SELECT emp_id, COUNT(*) AS sale_count,
SUM(amount) AS total_sales
    FROM sales
    GROUP BY emp_id
) s ON e.emp_id = s.emp_id
-- 左连接订单汇总（子查询）
LEFT JOIN (
    SELECT emp_id, COUNT(*) AS order_count
    FROM orders
    GROUP BY emp_id
) o ON e.emp_id = o.emp_id
ORDER BY total_sales DESC;

-- 步骤 3：查看结果
SELECT * FROM emp_performance_report;
```

예상 출력 (상위 5 행):

```
+--------+--------+------------+-------+----------+-----------+--------+----------+
```

预期输出（前 5 行）：

```
+-------+-------+----------+-------+----------+-----------+--------+----------+
```

```
| emp_id | emp_name | dept_name | salary |
sale_count | total_sales | order_count |
salary_level |
+--------+--------+-----------+-------+----------
+----------+--------+----------+
| E006   | 杨帆   | 财务管理部  | 14000 |          3
|    469000 |           5 | 중           |
| E015   | 朱峰   | 财务管理部  | 20000 |          2
|    385000 |           3 | 고           |
| E003   | 王强   | 技术研发部  | 18000 |          2
|    255000 |           2 | 고           |
| E002   | 李娜   | 市场营销部  | 12000 |          3
|    286000 |           4 | 중           |
...
+--------+--------+-----------+-------+----------
+----------+--------+----------+
```

종합 문제 2 참고 정답: 월별 판매 분석

```sql
SELECT
    SUBSTR(sale_date, 1, 7)       AS 월별,
    COUNT(*)                      AS 판매횟수,
    SUM(amount)                   AS 총판매액,
    MAX(amount)                   AS 최고단건금액,
    COUNT(DISTINCT product_type)  AS 제품종류수
FROM sales
GROUP BY SUBSTR(sale_date, 1, 7)
ORDER BY 월별;
```

예상 출력:

```
+---------+----------+----------+--------------+--
----------+
|  월별   | 판매횟수 | 총판매액 | 최고단건금액 |
제품종류수 |
+---------+----------+----------+--------------+--
----------+
| 2023-01 |        6 |   598000 |       145000 |
3 |
| 2023-02 |        7 |   802000 |       200000 |
3 |
| 2023-03 |        7 |   797000 |       185000 |
3 |
+---------+----------+----------+--------------+--
----------+
```

```
|emp_id |姓名    |部门名称  |月薪    |销售次数  |总销售
额   |订单数  |薪资等级 |
+-------+-------+----------+-------+----------+---
-------+--------+---------+
| E006  | 杨帆  | 财务管理部 | 14000 | 3        |
469000   | 5      | 中       |
| E015  | 朱峰  | 财务管理部 | 20000 | 2        |
385000   | 3      | 高       |
| E003  | 王强  | 技术研发部 | 18000 | 2        |
255000   | 2      | 高       |
| E002  | 李娜  | 市场营销部 | 12000 | 3        |
286000   | 4      | 中       |
...
+-------+-------+----------+-------+----------+---
-------+--------+---------+
```

综合题 2 参考答案：月度销售分析

```sql
SELECT
    SUBSTR(sale_date, 1, 7)   AS 月份,
    COUNT(*)                  AS 销售次数,
    SUM(amount)               AS 总销售额,
    MAX(amount)               AS 最高单笔金额,
    COUNT(DISTINCT product_type) AS 产品种类数
FROM sales
GROUP BY SUBSTR(sale_date, 1, 7)
ORDER BY 月份;
```

预期输出：

```
+---------+----------+----------+--------------+--
---------+
|  月份   | 销售次数 | 总销售额 | 最高单笔金额 | 产品
种类数 |
+---------+----------+----------+--------------+--
---------+
| 2023-01 | 6        | 598000   | 145000       | 3
|
| 2023-02 | 7        | 802000   | 200000       | 3
|
| 2023-03 | 7        | 797000   | 185000       | 3
|
+---------+----------+----------+--------------+--
---------+
```

확장: 월별 판매 왕 조회 (JOIN 사용) :

```sql
-- 월별 판매액 1 위 직원 조회
SELECT
    m.month_str   AS 월별,
    m.total_sales AS 월판매액,
    e.emp_name    AS 판매왕
FROM (
    SELECT
        SUBSTR(sale_date, 1, 7) AS month_str,
        emp_id,
        SUM(amount) AS total_sales,
        RANK() OVER (
            PARTITION BY SUBSTR(sale_date, 1, 7)
            ORDER BY SUM(amount) DESC
        ) AS rnk
    FROM sales
    GROUP BY SUBSTR(sale_date, 1, 7), emp_id
) m
INNER JOIN employees e ON m.emp_id = e.emp_id
WHERE m.rnk = 1
ORDER BY m.month_str;
```

종합 문제 3 참고 정답: 데이터 정제 및 변환

```sql
-- 단계 1: 대상 테이블 생성
CREATE TABLE IF NOT EXISTS below_avg_employees (
    emp_id          STRING  COMMENT '직원 번호',
    emp_name        STRING  COMMENT '직원 이름',
    salary          DOUBLE  COMMENT '개인 월급',
    dept_avg_salary DOUBLE  COMMENT '부서 평균 월급',
    salary_gap      DOUBLE  COMMENT '월급 차액 (음수) '
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 단계 2: 계산 후 데이터 삽입
INSERT OVERWRITE TABLE below_avg_employees
SELECT
    e.emp_id,
    e.emp_name,
    e.salary,
    ROUND(d_avg.avg_sal, 0)             AS dept_avg_salary,
    ROUND(e.salary - d_avg.avg_sal, 0)  AS salary_gap
FROM employees e
```

扩展：查询每月销售最高的员工（使用 JOIN）：

```sql
-- 先找出每月最高销售额
SELECT
    m.month_str,
    m.total_sales AS 月销售额,
    e.emp_name    AS 销售冠军
FROM (
    SELECT
        SUBSTR(sale_date, 1, 7) AS month_str,
        emp_id,
        SUM(amount) AS total_sales,
        RANK() OVER (
            PARTITION BY SUBSTR(sale_date, 1, 7)
            ORDER BY SUM(amount) DESC
        ) AS rnk
    FROM sales
    GROUP BY SUBSTR(sale_date, 1, 7), emp_id
) m
INNER JOIN employees e ON m.emp_id = e.emp_id
WHERE m.rnk = 1
ORDER BY m.month_str;
```

综合题 3 参考答案：数据清洗与转换

```sql
-- 步骤 1：创建目标表
CREATE TABLE IF NOT EXISTS below_avg_employees (
    emp_id          STRING  COMMENT '员工编号',
    emp_name        STRING  COMMENT '员工姓名',
    salary          DOUBLE  COMMENT '个人薪资',
    dept_avg_salary DOUBLE  COMMENT '部门平均薪资',
    salary_gap      DOUBLE  COMMENT '薪资差额（负数）'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 步骤 2：计算并插入数据
INSERT OVERWRITE TABLE below_avg_employees
SELECT
    e.emp_id,
    e.emp_name,
    e.salary,
    ROUND(d_avg.avg_sal, 0)             AS dept_avg_salary,
    ROUND(e.salary - d_avg.avg_sal, 0)  AS salary_gap
FROM employees e
```

```
INNER JOIN (
    SELECT dept_id, AVG(salary) AS avg_sal
    FROM employees
    GROUP BY dept_id
) d_avg ON e.dept_id = d_avg.dept_id
WHERE e.salary < d_avg.avg_sal
ORDER BY salary_gap ASC;

-- 단계 3: 결과 확인
SELECT * FROM below_avg_employees;

-- 단계 4: 로컬로 내보내기
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/below_avg'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT * FROM below_avg_employees;
```

종합 문제 4 참고 정답: Top N 분석 (윈도우 함수)

```sql
-- ROW_NUMBER() 윈도우 함수를 사용한 Top N 분석
SELECT
    region      AS 지역,
    emp_id      AS 직원번호,
    emp_name    AS 직원이름,
    dept_name   AS 소속부서,
    total_sales AS 총판매액
FROM (
    SELECT
        s.region,
        e.emp_id,
        e.emp_name,
        d.dept_name,
        SUM(s.amount) AS total_sales,
        ROW_NUMBER() OVER (
            PARTITION BY s.region
            ORDER BY SUM(s.amount) DESC
        ) AS rn
    FROM sales s
    INNER JOIN employees e   ON s.emp_id =
e.emp_id
    INNER JOIN departments d ON e.dept_id =
d.dept_id
    GROUP BY s.region, e.emp_id, e.emp_name,
d.dept_name
) t
WHERE t.rn <= 2
ORDER BY region, total_sales DESC;
```

예상 출력:

```
INNER JOIN (
    SELECT dept_id, AVG(salary) AS avg_sal
    FROM employees
    GROUP BY dept_id
) d_avg ON e.dept_id = d_avg.dept_id
WHERE e.salary < d_avg.avg_sal
ORDER BY salary_gap ASC;

-- 步骤 3：验证
SELECT * FROM below_avg_employees;

-- 步骤 4：导出到本地
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/below_avg'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT * FROM below_avg_employees;
```

综合题 4 参考答案：Top N 分析（窗口函数）

```sql
-- 使用 ROW_NUMBER() 窗口函数实现 Top N 分析
SELECT
    region      AS 区域,
    emp_id      AS 员工编号,
    emp_name    AS 员工姓名,
    dept_name   AS 所属部门,
    total_sales AS 区域总销售额
FROM (
    SELECT
        s.region,
        e.emp_id,
        e.emp_name,
        d.dept_name,
        SUM(s.amount) AS total_sales,
        ROW_NUMBER() OVER (
            PARTITION BY s.region
            ORDER BY SUM(s.amount) DESC
        ) AS rn
    FROM sales s
    INNER JOIN employees e   ON s.emp_id =
e.emp_id
    INNER JOIN departments d ON e.dept_id =
d.dept_id
    GROUP BY s.region, e.emp_id, e.emp_name,
d.dept_name
) t
WHERE t.rn <= 2
ORDER BY region, total_sales DESC;
```

预期输出：

```
+------+----------+----------+------------+-------
---+
| 지역 | 직원번호 | 직원이름 |  소속부서  | 총판매액 |
+------+----------+----------+------------+-------
---+
| 华北 | E003     | 王强     | 技术研发部 |
255000 |
| 华北 | E007     | 赵磊     | 技术研发部 |
197000 |
| 华南 | E006     | 杨帆     | 财务管理部 |
469000 |
| 华南 | E015     | 朱峰     | 财务管理部 |
385000 |
| 华东 | E002     | 李娜     | 市场营销部 |
286000 |
| 华东 | E005     | 陈静     | 市场营销部 |
225000 |
+------+----------+----------+------------+-------
---+
```

📊 채점 참고 기준

문제 유형	배점	채점 포인트
제 1 장 연습	15 점（각 5 점）	명령어 문법 정확, 데이터 행 수 일치
제 2 장 연습	20 점（각 5 점）	테이블 생성 정확, INSERT 결과 정확
제 3 장 연습	20 점（각 5 점）	WHERE 조건 정확, 결과 완전
제 4 장 연습	20 점（각 5 점）	집계 함수 사용 정확, HAVING 조건 정확
제 5 장 연습	20 점（각 5 점）	JOIN 유형 선택 정확, 조인 조건 정확
제 6 장 연습	15 점（각 5 점）	서브쿼리 논리 정확
제 7 장 연습	10 점（각 5 점）	내보내기 경로 및 형식 정확
제 8 장 종합	40 점（각 10 점）	로직 완성, 결과

```
+------+--------+--------+----------+----------+
| 区域 |员工编号|员工姓名| 所属部门 |总销售额  |
+------+--------+--------+----------+----------+
| 华北 | E003   | 王强   | 技术研发部 | 255000 |
| 华北 | E007   | 赵磊   | 技术研发部 | 197000 |
| 华南 | E006   | 杨帆   | 财务管理部 | 469000 |
| 华南 | E015   | 朱峰   | 财务管理部 | 385000 |
| 华东 | E002   | 李娜   | 市场营销部 | 286000 |
| 华东 | E005   | 陈静   | 市场营销部 | 225000 |
+------+--------+--------+----------+----------+
```

📊 评分参考标准

题型	分值	评分要点
第 1 章 练习	15 分（各 5 分）	命令语法正确、数据行数符合预期
第 2 章 练习	20 分（各 5 分）	表创建正确、INSERT 结果准确
第 3 章 练习	20 分（各 5 分）	WHERE 条件正确、结果完整
第 4 章 练习	20 分（各 5 分）	聚合函数使用正确、HAVING 条件正确
第 5 章 练习	20 分（各 5 分）	JOIN 类型选择正确、连接条件正确
第 6 章 练习	15 分（各 5 分）	子查询逻辑正确
第 7 章 练习	10 分（各 5 分）	导出路径和格式正确
第 8 章 综合	40 分（各 10 分）	逻辑完整、结果

정확 |

참고 정답 문서 끝

> 📌 **교수법 팁**: 종합 문제 4 는 윈도우 함수를 사용합니다. 학생이 아직 이 내용을 배우지 않았다면, 서브쿼리 + LIMIT 을 사용하는 근사 해법도 인정할 수 있습니다.

准确 |

参考答案文档结束

> 📌 **教学提示**：综合题 4 涉及窗口函数，如学生尚未学习该内容，可接受使用子查询 + LIMIT 的近似解法。