Hive 제 5 장 DML 연산 실습 워크북

> **과목**: 빅데이터 기술과 응용
> **단원**: 제 5 장 Hive DML（데이터 조작 언어）
> **실습 시간**: 약 4 시간
> **데이터베이스**: `unit5db`
> **데이터 디렉터리**: `/usr/datas/unit5/`
> **접속 도구**: `hive-bee`（beeline 별칭）
> **난이도**: ★★★☆☆

📋 워크북 목차

번호	단원명	예상 시간	섹션
제 0 장	실험 환경 준비 및 데이터셋 초기화	20 분	§0
제 1 장	LOAD DATA 데이터 로드 작업	30 분	§1
제 2 장	INSERT 데이터 삽입 작업	40 분	§2
제 3 장	SELECT 기본 조회	30 분	§3
제 4 장	집계 조회 및 그룹 통계（GROUP BY / HAVING）	30 분	§4
제 5 장	다중 테이블 조인 조회（JOIN）	35 분	§5
제 6 장	서브쿼리 및 중첩 쿼리	20 분	§6
제 7 장	데이터 내보내기（INSERT OVERWRITE DIRECTORY）	15 분	§7
제 8 장	종합 연습문제	40 분	§8

📐 제 0 장 실험 환경 준비 및 데이터셋 초기화

학습 목표

Hive 第 5 章 DML 操作实践工作手册

> **课程**：大数据技术与应用
> **章节**：第 5 章 Hive DML（数据操作语言）
> **实验时长**：约 4 小时
> **数据库**：`unit5db`
> **数据目录**：`/usr/datas/unit5/`
> **连接工具**：`hive-bee`（beeline 别名）
> **难度**：★★★☆☆

📋 工作手册目录

序号	章节名称	预计时间	页码
第 0 章	实验环境准备与数据集初始化	20 分钟	§0
第 1 章	LOAD DATA 数据加载操作	30 分钟	§1
第 2 章	INSERT 数据插入操作	40 分钟	§2
第 3 章	SELECT 基础查询	30 分钟	§3
第 4 章	聚合查询与分组统计（GROUP BY / HAVING）	30 分钟	§4
第 5 章	多表连接查询（JOIN）	35 分钟	§5
第 6 章	子查询与嵌套查询	20 分钟	§6
第 7 章	数据导出操作（INSERT OVERWRITE DIRECTORY）	15 分钟	§7
第 8 章	综合练习题	40 分钟	§8

📐 第 0 章 实验环境准备与数据集初始化

学习目标

- Hive 서비스가 정상적으로 실행 중인지 확인
- 이번 실험에 필요한 데이터베이스와 테이블 구조 생성
- 실험용 데이터 파일 준비

0.1 Beeline 실행 및 Hive 연결

터미널을 열고 다음 명령어를 입력하여 Beeline 을 시작합니다:

```bash
hive-bee
```

연결에 성공하면 다음과 같은 프롬프트가 나타납니다:

```
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.3)
0: jdbc:hive2://localhost:10000>
```

> ⚠️ **주의**: 연결에 실패하면 HiveServer2 서비스가 실행 중인지 확인하세요.

0.2 실험 데이터베이스 생성 및 전환

```sql
-- 실험 데이터베이스 생성 (이미 존재하면 건너뜀)
CREATE DATABASE IF NOT EXISTS unit5db
COMMENT 'Hive DML 실습 실험 데이터베이스'
LOCATION '/user/hive/warehouse/unit5db';

-- 실험 데이터베이스로 전환
USE unit5db;

-- 현재 데이터베이스 확인
SELECT current_database();
```

- 确认 Hive 服务正常运行
- 创建本次实验所需数据库和表结构
- 准备实验用数据文件

0.1 启动 Beeline 并连接 Hive

打开终端，输入以下命令启动 Beeline：

```bash
hive-bee
```

连接成功后，您将看到类似如下的提示符：

```
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.3)
0: jdbc:hive2://localhost:10000>
```

> ⚠️ **注意**：如果连接失败，请检查 HiveServer2 服务是否已启动。

0.2 创建并切换到实验数据库

```sql
-- 创建实验数据库（如果已存在则跳过）
CREATE DATABASE IF NOT EXISTS unit5db
COMMENT 'Hive DML 实践实验数据库'
LOCATION '/user/hive/warehouse/unit5db';

-- 切换到实验数据库
USE unit5db;

-- 确认当前数据库
SELECT current_database();
```

0.3 데이터 파일 준비

> 📌 이번 실험의 모든 데이터 파일은 `/usr/datas/unit5/` 디렉터리에 미리 준비되어 있습니다.
> 터미널（Beeline 아님）에서 다음 명령어로 파일 존재 여부를 확인하세요:

새 터미널 창을 열고 다음 명령어를 실행합니다:

```bash
# 데이터 파일 디렉터리 확인
ls -la /usr/datas/unit5/
```

다음 파일들이 보여야 합니다:

```
departments.txt
employees.txt
sales.txt
orders.txt
order_items.txt
```

0.4 데이터 파일 생성（파일이 없을 경우, 교사가 실행하는 단계）

> 📌 아래 명령어는 교사가 수업 전에 준비하는 내용입니다. 학생은 데이터 구조를 파악하는 참고용으로 확인하세요.

```bash
# 데이터 디렉터리 생성
sudo mkdir -p /usr/datas/unit5

# 부서 데이터 파일 departments.txt 생성
# 형식: dept_id, dept_name, location, budget
cat > /usr/datas/unit5/departments.txt << 'EOF'
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
```

0.3 准备数据文件

> 📌 本次实验所有数据文件已预先放置在 `/usr/datas/unit5/` 目录下。

> 请在终端（非 Beeline）中执行以下命令确认文件存在：

打开新的终端窗口，执行以下命令：

```bash
# 查看数据文件目录
ls -la /usr/datas/unit5/
```

您应该看到以下文件：

```
departments.txt
employees.txt
sales.txt
orders.txt
order_items.txt
```

0.4 创建数据文件（如文件不存在，由老师执行此步骤）

> 📌 以下命令由教师在课前准备好，学生可参考了解数据结构。

```bash
# 创建数据目录
sudo mkdir -p /usr/datas/unit5

# 创建部门数据文件 departments.txt
# 格式：dept_id, dept_name, location, budget
cat > /usr/datas/unit5/departments.txt << 'EOF'
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
EOF

# 직원 데이터 파일 employees.txt 생성
# 형식: emp_id, emp_name, age, gender, dept_id, salary, hire_date, city
cat > /usr/datas/unit5/employees.txt << 'EOF'
E001,张伟,32,男,D001,15000,2019-03-15,北京
E002,李娜,28,女,D002,12000,2020-06-01,上海
E003,王强,35,男,D001,18000,2018-01-10,北京
E004,刘洋,26,女,D003,9000,2021-08-20,北京
E005,陈静,30,女,D002,13000,2019-11-05,上海
E006,杨帆,33,男,D004,14000,2018-07-22,广州
E007,赵磊,29,男,D001,16000,2020-02-14,北京
E008,周梅,27,女,D005,10000,2021-03-30,深圳
E009,吴刚,38,男,D006,17000,2017-09-08,杭州
E010,郑芳,31,女,D003,11000,2019-05-17,北京
E011,孙浩,24,男,D005,8500,2022-01-10,深圳
E012,马丽,36,女,D006,16500,2018-04-25,杭州
E013,胡涛,28,男,D002,12500,2020-10-12,上海
E014,林晓,29,女,D001,14500,2020-08-03,北京
E015,朱峰,41,男,D004,20000,2016-12-01,广州
EOF

# 판매 데이터 파일 sales.txt 생성
# 형식: sale_id, emp_id, sale_date, amount, product_type, region
cat > /usr/datas/unit5/sales.txt << 'EOF'
S001,E002,2023-01-05,85000,软件产品,华东
S002,E005,2023-01-12,62000,硬件设备,华东
S003,E013,2023-01-20,43000,软件产品,华东
S004,E002,2023-02-03,91000,软件产品,华东
S005,E005,2023-02-15,75000,服务项目,华东
S006,E013,2023-02-28,55000,硬件设备,华东
S007,E002,2023-03-10,110000,软件产品,华东
S008,E005,2023-03-22,88000,软件产品,华东
S009,E013,2023-03-31,67000,服务项目,华东
S010,E003,2023-01-08,120000,软件产品,华北
S011,E007,2023-01-25,95000,硬件设备,华北
S012,E014,2023-02-10,78000,软件产品,华北
S013,E003,2023-02-20,135000,软件产品,华北
S014,E007,2023-03-05,102000,服务项目,华北
S015,E014,2023-03-18,89000,硬件设备,华北
S016,E006,2023-01-15,145000,软件产品,华南
S017,E015,2023-02-08,200000,服务项目,华南
S018,E006,2023-02-25,168000,硬件设备,华南
S019,E015,2023-03-12,185000,软件产品,华南
S020,E006,2023-03-28,156000,软件产品,华南
```

```
D006,产品设计部,杭州,700000
EOF

# 创建员工数据文件 employees.txt
# 格式：emp_id, emp_name, age, gender, dept_id, salary, hire_date, city
cat > /usr/datas/unit5/employees.txt << 'EOF'
E001,张伟,32,男,D001,15000,2019-03-15,北京
E002,李娜,28,女,D002,12000,2020-06-01,上海
E003,王强,35,男,D001,18000,2018-01-10,北京
E004,刘洋,26,女,D003,9000,2021-08-20,北京
E005,陈静,30,女,D002,13000,2019-11-05,上海
E006,杨帆,33,男,D004,14000,2018-07-22,广州
E007,赵磊,29,男,D001,16000,2020-02-14,北京
E008,周梅,27,女,D005,10000,2021-03-30,深圳
E009,吴刚,38,男,D006,17000,2017-09-08,杭州
E010,郑芳,31,女,D003,11000,2019-05-17,北京
E011,孙浩,24,男,D005,8500,2022-01-10,深圳
E012,马丽,36,女,D006,16500,2018-04-25,杭州
E013,胡涛,28,男,D002,12500,2020-10-12,上海
E014,林晓,29,女,D001,14500,2020-08-03,北京
E015,朱峰,41,男,D004,20000,2016-12-01,广州
EOF

# 创建销售数据文件 sales.txt
# 格式：sale_id, emp_id, sale_date, amount, product_type, region
cat > /usr/datas/unit5/sales.txt << 'EOF'
S001,E002,2023-01-05,85000,软件产品,华东
S002,E005,2023-01-12,62000,硬件设备,华东
S003,E013,2023-01-20,43000,软件产品,华东
S004,E002,2023-02-03,91000,软件产品,华东
S005,E005,2023-02-15,75000,服务项目,华东
S006,E013,2023-02-28,55000,硬件设备,华东
S007,E002,2023-03-10,110000,软件产品,华东
S008,E005,2023-03-22,88000,软件产品,华东
S009,E013,2023-03-31,67000,服务项目,华东
S010,E003,2023-01-08,120000,软件产品,华北
S011,E007,2023-01-25,95000,硬件设备,华北
S012,E014,2023-02-10,78000,软件产品,华北
S013,E003,2023-02-20,135000,软件产品,华北
S014,E007,2023-03-05,102000,服务项目,华北
S015,E014,2023-03-18,89000,硬件设备,华北
S016,E006,2023-01-15,145000,软件产品,华南
S017,E015,2023-02-08,200000,服务项目,华南
S018,E006,2023-02-25,168000,硬件设备,华南
S019,E015,2023-03-12,185000,软件产品,华南
S020,E006,2023-03-28,156000,软件产品,华南
```

```
EOF

# 주문 데이터 파일 orders.txt 생성
# 형식: order_id, customer_id, order_date, total_amount, status, emp_id
cat > /usr/datas/unit5/orders.txt << 'EOF'
O1001,C001,2023-01-05,85000,已完成,E002
O1002,C002,2023-01-12,62000,已完成,E005
O1003,C003,2023-01-20,43000,已取消,E013
O1004,C001,2023-02-03,91000,已完成,E002
O1005,C004,2023-02-15,75000,已完成,E005
O1006,C005,2023-02-28,55000,处理中,E013
O1007,C002,2023-03-10,110000,已完成,E002
O1008,C003,2023-03-22,88000,已完成,E005
O1009,C006,2023-03-31,67000,处理中,E013
O1010,C007,2023-01-08,120000,已完成,E003
O1011,C008,2023-01-25,95000,已完成,E007
O1012,C009,2023-02-10,78000,已取消,E014
O1013,C007,2023-02-20,135000,已完成,E003
O1014,C010,2023-03-05,102000,已完成,E007
O1015,C001,2023-03-18,89000,处理中,E014
O1016,C011,2023-01-15,145000,已完成,E006
O1017,C012,2023-02-08,200000,已完成,E015
O1018,C011,2023-02-25,168000,已完成,E006
O1019,C013,2023-03-12,185000,已完成,E015
O1020,C014,2023-03-28,156000,已完成,E006
EOF

# 주문 상세 데이터 파일 order_items.txt 생성
# 형식: item_id, order_id, product_name, quantity, unit_price
cat > /usr/datas/unit5/order_items.txt << 'EOF'
I001,O1001,企业管理软件,2,30000
I002,O1001,技术支持服务,1,25000
I003,O1002,服务器设备,1,50000
I004,O1002,网络交换机,2,6000
I005,O1003,数据分析软件,1,43000
I006,O1004,企业管理软件,3,30000
I007,O1004,培训服务,1,1000
I008,O1005,云服务套餐,1,75000
I009,O1006,打印机,3,15000
I010,O1006,墨盒套装,10,1000
I011,O1007,企业管理软件,2,30000
I012,O1007,移动端授权,10,5000
I013,O1008,数据分析软件,2,43000
I014,O1008,技术支持,1,2000
I015,O1009,云服务套餐,1,67000
EOF

# 파일 권한 설정
sudo chmod 644 /usr/datas/unit5/*.txt
echo "데이터 파일 준비 완료！"
```

```
EOF

# 创建订单数据文件 orders.txt
# 格式：order_id, customer_id, order_date, total_amount, status, emp_id
cat > /usr/datas/unit5/orders.txt << 'EOF'
O1001,C001,2023-01-05,85000,已完成,E002
O1002,C002,2023-01-12,62000,已完成,E005
O1003,C003,2023-01-20,43000,已取消,E013
O1004,C001,2023-02-03,91000,已完成,E002
O1005,C004,2023-02-15,75000,已完成,E005
O1006,C005,2023-02-28,55000,处理中,E013
O1007,C002,2023-03-10,110000,已完成,E002
O1008,C003,2023-03-22,88000,已完成,E005
O1009,C006,2023-03-31,67000,处理中,E013
O1010,C007,2023-01-08,120000,已完成,E003
O1011,C008,2023-01-25,95000,已完成,E007
O1012,C009,2023-02-10,78000,已取消,E014
O1013,C007,2023-02-20,135000,已完成,E003
O1014,C010,2023-03-05,102000,已完成,E007
O1015,C001,2023-03-18,89000,处理中,E014
O1016,C011,2023-01-15,145000,已完成,E006
O1017,C012,2023-02-08,200000,已完成,E015
O1018,C011,2023-02-25,168000,已完成,E006
O1019,C013,2023-03-12,185000,已完成,E015
O1020,C014,2023-03-28,156000,已完成,E006
EOF

# 创建订单明细数据文件 order_items.txt
# 格式：item_id, order_id, product_name, quantity, unit_price
cat > /usr/datas/unit5/order_items.txt << 'EOF'
I001,O1001,企业管理软件,2,30000
I002,O1001,技术支持服务,1,25000
I003,O1002,服务器设备,1,50000
I004,O1002,网络交换机,2,6000
I005,O1003,数据分析软件,1,43000
I006,O1004,企业管理软件,3,30000
I007,O1004,培训服务,1,1000
I008,O1005,云服务套餐,1,75000
I009,O1006,打印机,3,15000
I010,O1006,墨盒套装,10,1000
I011,O1007,企业管理软件,2,30000
I012,O1007,移动端授权,10,5000
I013,O1008,数据分析软件,2,43000
I014,O1008,技术支持,1,2000
I015,O1009,云服务套餐,1,67000
EOF

# 设置文件权限
sudo chmod 644 /usr/datas/unit5/*.txt
echo "数据文件准备完成！"
```

```
ls -la /usr/datas/unit5/
```

0.5 실험용 테이블 구조 생성

Beeline 에서 다음 테이블 생성 구문을 실행합니다:

```sql
USE unit5db;

-- ① 부서 테이블 생성
CREATE TABLE IF NOT EXISTS departments (
    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
TBLPROPERTIES ('skip.header.line.count'='0');

-- ② 직원 테이블 생성
CREATE TABLE IF NOT EXISTS employees (
    emp_id      STRING      COMMENT '직원 번호',
    emp_name    STRING      COMMENT '직원 이름',
    age         INT         COMMENT '나이',
    gender      STRING      COMMENT '성별',
    dept_id     STRING      COMMENT '소속 부서
번호',
    salary      DOUBLE      COMMENT '월급 (원) ',
    hire_date   STRING      COMMENT '입사일',
    city        STRING      COMMENT '소재 도시'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- ③ 판매 기록 테이블 생성
CREATE TABLE IF NOT EXISTS sales (
    sale_id         STRING      COMMENT '판매
번호',
    emp_id          STRING      COMMENT '판매 직원
번호',
    sale_date       STRING      COMMENT '판매
```


```
ls -la /usr/datas/unit5/
```

0.5 创建实验所需表结构

在 Beeline 中执行以下建表语句：

```sql
USE unit5db;

-- ① 创建部门表
CREATE TABLE IF NOT EXISTS departments (
    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
TBLPROPERTIES ('skip.header.line.count'='0');

-- ② 创建员工表
CREATE TABLE IF NOT EXISTS employees (
    emp_id      STRING      COMMENT '员工编号',
    emp_name    STRING      COMMENT '员工姓名',
    age         INT         COMMENT '年龄',
    gender      STRING      COMMENT '性别',
    dept_id     STRING      COMMENT '所属部门编号',
    salary      DOUBLE      COMMENT '月薪（元）',
    hire_date   STRING      COMMENT '入职日期',
    city        STRING      COMMENT '所在城市'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- ③ 创建销售记录表
CREATE TABLE IF NOT EXISTS sales (
    sale_id         STRING      COMMENT '销售编号',
    emp_id          STRING      COMMENT '销售员工编
号',
    sale_date       STRING      COMMENT '销售日期',
```

```
날짜',
    amount          DOUBLE      COMMENT '판매 금액
(원)',
    product_type    STRING      COMMENT '제품
유형',
    region          STRING      COMMENT '판매 지역'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- ④ 주문 테이블 생성
CREATE TABLE IF NOT EXISTS orders (
    order_id        STRING      COMMENT '주문
번호',
    customer_id     STRING      COMMENT '고객
번호',
    order_date      STRING      COMMENT '주문
날짜',
    total_amount    DOUBLE      COMMENT '주문
총금액(원)',
    status          STRING      COMMENT '주문
상태',
    emp_id          STRING      COMMENT '담당 직원
번호'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- ⑤ 주문 상세 테이블 생성
CREATE TABLE IF NOT EXISTS order_items (
    item_id         STRING      COMMENT '상세
번호',
    order_id        STRING      COMMENT '주문
번호',
    product_name    STRING      COMMENT '제품명',
    quantity        INT         COMMENT '수량',
    unit_price      DOUBLE      COMMENT '단가(원)
'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 테이블 생성 확인
SHOW TABLES;
```

예상 출력:

```
    amount          DOUBLE      COMMENT '销售金额
(元)',
    product_type    STRING      COMMENT '产品类型',
    region          STRING      COMMENT '销售区域'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- ④ 创建订单表
CREATE TABLE IF NOT EXISTS orders (
    order_id        STRING      COMMENT '订单编号',
    customer_id     STRING      COMMENT '客户编号',
    order_date      STRING      COMMENT '订单日期',
    total_amount    DOUBLE      COMMENT '订单总金额
(元)',
    status          STRING      COMMENT '订单状态',
    emp_id          STRING      COMMENT '负责员工编
号'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- ⑤ 创建订单明细表
CREATE TABLE IF NOT EXISTS order_items (
    item_id         STRING      COMMENT '明细编号',
    order_id        STRING      COMMENT '订单编号',
    product_name    STRING      COMMENT '产品名称',
    quantity        INT         COMMENT '数量',
    unit_price      DOUBLE      COMMENT '单价(元)
'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 验证表已创建
SHOW TABLES;
```

预期输出：

```
+--------------+
|   tab_name   |
+--------------+
| departments  |
| employees    |
| order_items  |
| orders       |
| sales        |
+--------------+
```

📥 제 1 장 LOAD DATA 데이터 로드 작업

학습 목표

- `LOAD DATA LOCAL INPATH` 명령어 사용법 습득
- `OVERWRITE`와 추가 로드의 차이 이해
- 데이터가 성공적으로 로드되었는지 검증하는 방법 습득

1.1 LOAD DATA 기본 문법

```sql
-- 기본 문법
LOAD DATA [LOCAL] INPATH '파일경로'
[OVERWRITE] INTO TABLE 테이블명
[PARTITION (파티션컬럼 1=값 1, 파티션컬럼 2=값 2 ...)];
```

키워드	설명
`LOCAL`	로컬 파일 시스템에서 로드 (LOCAL 없으면 HDFS 에서 로드)
`OVERWRITE`	테이블의 기존 데이터를 덮어씀
`INPATH`	데이터 파일 경로 지정

1.2 예제 실습: 부서 데이터 로드

```sql
```


```
+--------------+
|   tab_name   |
+--------------+
| departments  |
| employees    |
| order_items  |
| orders       |
| sales        |
+--------------+
```

📥 第 1 章 LOAD DATA 数据加载操作

学习目标

- 掌握 `LOAD DATA LOCAL INPATH` 命令的使用
- 理解 `OVERWRITE` 与追加加载的区别
- 学会验证数据是否成功加载

1.1 LOAD DATA 基本语法

```sql
-- 基本语法
LOAD DATA [LOCAL] INPATH 'filepath'
[OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)];
```

关键字	说明
`LOCAL`	从本地文件系统加载（不加 LOCAL 则从 HDFS 加载）
`OVERWRITE`	覆盖表中现有数据
`INPATH`	指定数据文件路径

1.2 实例演示：加载部门数据

```sql
```

```sql
USE unit5db;

-- 부서 데이터 로드 (로컬 파일 시스템에서)
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
INTO TABLE departments;

-- 로드 결과 확인
SELECT * FROM departments;
```

예상 결과:

```
+----------+------------+--------+---------+
| 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  |
+----------+------------+--------+---------+
```

1.3 예제 실습: 직원 데이터 로드

```sql
-- 직원 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/employees.txt'
INTO TABLE employees;

-- 로드 결과 확인 (상위 5 행)
SELECT * FROM employees LIMIT 5;

-- 전체 행 수 집계
SELECT COUNT(*) AS total_employees FROM employees;
```

1.4 예제 실습: 판매·주문 데이터 로드

```sql
-- 판매 데이터 로드
LOAD DATA LOCAL INPATH '/usr/datas/unit5/sales.txt'
INTO TABLE sales;

-- 주문 데이터 로드
```

```sql
USE unit5db;

-- 加载部门数据（从本地文件系统）
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
INTO TABLE departments;

-- 查看加载结果
SELECT * FROM departments;
```

预期结果：

```
+----------+------------+--------+---------+
| 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  |
+----------+------------+--------+---------+
```

1.3 实例演示：加载员工数据

```sql
-- 加载员工数据
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/employees.txt'
INTO TABLE employees;

-- 查看加载结果（前 5 条）
SELECT * FROM employees LIMIT 5;

-- 统计总行数
SELECT COUNT(*) AS total_employees FROM employees;
```

1.4 实例演示：加载其他表数据

```sql
-- 加载销售数据
LOAD DATA LOCAL INPATH '/usr/datas/unit5/sales.txt'
INTO TABLE sales;

-- 加载订单数据
```

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/orders.txt'
INTO TABLE orders;

-- 주문 상세 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/order_items.txt'
INTO TABLE order_items;

-- 모든 테이블의 데이터 행 수 확인
SELECT 'departments' AS tbl, COUNT(*) AS cnt FROM
departments
UNION ALL
SELECT 'employees',   COUNT(*) FROM employees
UNION ALL
SELECT 'sales',       COUNT(*) FROM sales
UNION ALL
SELECT 'orders',      COUNT(*) FROM orders
UNION ALL
SELECT 'order_items', COUNT(*) FROM order_items;
```

예상 출력:

```
+--------------+-----+
|     tbl      | cnt |
+--------------+-----+
| departments  |   6 |
| employees    |  15 |
| sales        |  20 |
| orders       |  20 |
| order_items  |  15 |
+--------------+-----+
```

1.5 OVERWRITE 덮어쓰기 로드 실습

```sql
-- 생성 테스트 테이블
CREATE TABLE IF NOT EXISTS test_load (
    id     INT,
    name   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 같은 파일을 한 번 더 로드 (추가)
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
INTO TABLE departments;

-- 행 수 확인 (두 번 로드했으므로 12 행이어야 함)
SELECT COUNT(*) AS cnt FROM departments;

-- OVERWRITE 를 사용해 재로드 (최신 6 행만 유지)
```


```
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/orders.txt'
INTO TABLE orders;

-- 加载订单明细数据
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/order_items.txt'
INTO TABLE order_items;

-- 验证所有表的数据行数
SELECT 'departments' AS tbl, COUNT(*) AS cnt FROM
departments
UNION ALL
SELECT 'employees', COUNT(*) FROM employees
UNION ALL
SELECT 'sales', COUNT(*) FROM sales
UNION ALL
SELECT 'orders', COUNT(*) FROM orders
UNION ALL
SELECT 'order_items', COUNT(*) FROM order_items;
```

예상 출력:

```
+--------------+-----+
|     tbl      | cnt |
+--------------+-----+
| departments  |   6 |
| employees    |  15 |
| sales        |  20 |
| orders       |  20 |
| order_items  |  15 |
+--------------+-----+
```

1.5 OVERWRITE 覆盖加载演示

```sql
-- 创建测试表
CREATE TABLE IF NOT EXISTS test_load (
    id     INT,
    name   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 先加载一次数据（追加）
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
INTO TABLE departments;

-- 查看行数（应该有 12 行，因为加载了两次）
SELECT COUNT(*) AS cnt FROM departments;

-- 使用 OVERWRITE 重新加载（只保留最新的 6 行）
```

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
OVERWRITE INTO TABLE departments;

-- 다시 행 수 확인 (6 행으로 복원되어야 함)
SELECT COUNT(*) AS cnt FROM departments;
```

> 💡 **정리**:

> - `OVERWRITE` 없이 로드: 새 데이터가 기존 데이터에 추가됨 (기존 데이터 삭제 안 됨)

> - `OVERWRITE` 사용: 테이블을 먼저 비운 후 새 데이터를 로드함

🖊 제 1 장 연습문제

연습 1-1: `OVERWRITE` 방식으로 `employees.txt`를 `employees` 테이블에 다시 로드하여 테이블에 정확히 15 개의 레코드만 남도록 하세요.

연습 1-2: `sales` 테이블의 데이터 건수를 확인하고, 판매 금액이 가장 큰 상위 3 건의 레코드를 조회하세요 (`ORDER BY`와 `LIMIT` 사용).

연습 1-3: `dept_backup`이라는 새 테이블을 생성하세요 (구조는 `departments`와 동일). 그런 다음 `departments.txt` 데이터를 그 테이블에 로드하세요.

📝 제 2 장 INSERT 데이터 삽입 작업

학습 목표

- `INSERT INTO` 추가 삽입 습득

- `INSERT OVERWRITE` 덮어쓰기 삽입 습득

```
LOAD DATA LOCAL INPATH
'/usr/datas/unit5/departments.txt'
OVERWRITE INTO TABLE departments;

-- 再次查看行数（应该恢复为 6 行）
SELECT COUNT(*) AS cnt FROM departments;
```

> 💡 **小结**：

> - 不加 `OVERWRITE`：新数据追加到表中（不删除旧数据）

> - 加 `OVERWRITE`：先清空表，再加载新数据

🖊 第 1 章 练习题

练习 1-1：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。

练习 1-2：验证 `sales` 表中的数据条数，并查看销售金额最大的 3 条记录（使用 `ORDER BY` 和 `LIMIT`）。

练习 1-3：创建一个名为 `dept_backup` 的新表，表结构与 `departments` 相同，然后将 `departments.txt` 数据加载到该表中。

📝 第 2 章 INSERT 数据插入操作

学习目标

- 掌握 `INSERT INTO` 追加插入

- 掌握 `INSERT OVERWRITE` 覆盖插入

- `INSERT INTO TABLE ... SELECT` 쿼리 기반 삽입 습득
- 다중 테이블 삽입 (FROM ... INSERT INTO) 방법 이해

2.1 INSERT INTO VALUES 문법

```sql
-- 문법 형식
INSERT INTO TABLE 테이블명 VALUES (값1, 값2, ...);
```

```sql
USE unit5db;

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

-- 단일 행 삽입
INSERT INTO TABLE emp_test VALUES ('T001',
'테스트사용자A', 10000, '기술부');


-- 결과 확인
SELECT * FROM emp_test;
```

2.2 INSERT INTO 추가 삽입 (다중 행)

```sql
-- 여러 행을 한 번에 삽입
INSERT INTO TABLE emp_test VALUES
    ('T002', '테스트사용자B', 12000, '영업부'),
    ('T003', '테스트사용자C', 9500,  '인사부'),
    ('T004', '테스트사용자D', 15000, '기술부'),
    ('T005', '테스트사용자E', 11000, '재무부');

-- 모든 데이터 조회
```

- 学会使用 `INSERT INTO ... SELECT` 从查询结果插入
- 了解多表插入 (FROM ... INSERT INTO)

2.1 INSERT INTO VALUES 语法

```sql
-- 语法格式
INSERT INTO TABLE tablename VALUES (val1,
val2, ...);
```

```sql
USE unit5db;

-- 创建一个简单的测试表
CREATE TABLE IF NOT EXISTS emp_test (
    emp_id    STRING,
    emp_name  STRING,
    salary    DOUBLE,
    dept      STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 插入单条数据
INSERT INTO TABLE emp_test VALUES ('T001', '测试用
户A', 10000, '技术部');


-- 查看结果
SELECT * FROM emp_test;
```

2.2 INSERT INTO 追加插入 (多条数据)

```sql
-- 一次插入多条数据
INSERT INTO TABLE emp_test VALUES
    ('T002', '测试用户B', 12000, '市场部'),
    ('T003', '测试用户C', 9500,  '人事部'),
    ('T004', '测试用户D', 15000, '技术部'),
    ('T005', '测试用户E', 11000, '财务部');

-- 查看所有数据
```

```
SELECT * FROM emp_test;

-- 행 수 집계
SELECT COUNT(*) AS total FROM emp_test;
```

2.3 INSERT INTO ... SELECT (쿼리 결과로 삽입)

```sql
-- 고임금 직원 테이블 생성 (월급 >= 15000)
CREATE TABLE IF NOT EXISTS high_salary_emp (
    emp_id    STRING,
    emp_name  STRING,
    salary    DOUBLE,
    dept_id   STRING,
    city      STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- employees 테이블의 고임금 직원 데이터를 새 테이블에 삽입
INSERT INTO TABLE high_salary_emp
SELECT emp_id, emp_name, salary, dept_id, city
FROM employees
WHERE salary >= 15000;

-- 삽입 결과 확인
SELECT * FROM high_salary_emp ORDER BY salary DESC;
```

2.4 INSERT OVERWRITE ... SELECT (덮어쓰기 삽입)

```sql
-- high_salary_emp 데이터 갱신 (기준을 13000 으로 변경)
INSERT OVERWRITE TABLE high_salary_emp
SELECT emp_id, emp_name, salary, dept_id, city
FROM employees
WHERE salary >= 13000;

-- 갱신 후 결과 확인
SELECT * FROM high_salary_emp ORDER BY salary DESC;
SELECT COUNT(*) AS total FROM high_salary_emp;
```

```
SELECT * FROM emp_test;

-- 统计行数
SELECT COUNT(*) AS total FROM emp_test;
```

2.3 INSERT INTO ... SELECT (从查询结果插入)

```sql
-- 创建高薪员工表 (月薪 >= 15000)
CREATE TABLE IF NOT EXISTS high_salary_emp (
    emp_id    STRING,
    emp_name  STRING,
    salary    DOUBLE,
    dept_id   STRING,
    city      STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 将 employees 表中高薪员工数据插入新表
INSERT INTO TABLE high_salary_emp
SELECT emp_id, emp_name, salary, dept_id, city
FROM employees
WHERE salary >= 15000;

-- 验证插入结果
SELECT * FROM high_salary_emp ORDER BY salary DESC;
```

2.4 INSERT OVERWRITE ... SELECT (覆盖插入)

```sql
-- 假设需要更新 high_salary_emp 的数据 (将门槛改为 13000)
INSERT OVERWRITE TABLE high_salary_emp
SELECT emp_id, emp_name, salary, dept_id, city
FROM employees
WHERE salary >= 13000;

-- 验证更新后的结果
SELECT * FROM high_salary_emp ORDER BY salary DESC;
SELECT COUNT(*) AS total FROM high_salary_emp;
```

2.5 INSERT INTO ... SELECT 데이터 집계 사례

```sql
-- 부서별 급여 통계 테이블 생성
CREATE TABLE IF NOT EXISTS dept_salary_stats (
    dept_id       STRING,
    emp_count     INT,
    avg_salary    DOUBLE,
    max_salary    DOUBLE,
    min_salary    DOUBLE,
    total_salary  DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 통계 결과를 테이블에 삽입
INSERT OVERWRITE TABLE dept_salary_stats
SELECT
    dept_id,
    COUNT(*)     AS emp_count,
    AVG(salary)  AS avg_salary,
    MAX(salary)  AS max_salary,
    MIN(salary)  AS min_salary,
    SUM(salary)  AS total_salary
FROM employees
GROUP BY dept_id;

-- 통계 결과 확인
SELECT * FROM dept_salary_stats ORDER BY avg_salary
DESC;
```

2.6 FROM 절 다중 대상 삽입

```sql
-- 남직원.여직원을 각각 별도 테이블에 저장
CREATE TABLE IF NOT EXISTS male_employees (
    emp_id    STRING,
    emp_name  STRING,
    age       INT,
    salary    DOUBLE,
    dept_id   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE IF NOT EXISTS female_employees (
    emp_id    STRING,
    emp_name  STRING,
    age       INT,
    salary    DOUBLE,
    dept_id   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

2.5 INSERT INTO ... SELECT 数据汇总案例

```sql
-- 创建部门薪资统计表
CREATE TABLE IF NOT EXISTS dept_salary_stats (
    dept_id       STRING,
    emp_count     INT,
    avg_salary    DOUBLE,
    max_salary    DOUBLE,
    min_salary    DOUBLE,
    total_salary  DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

-- 将统计结果插入表中
INSERT OVERWRITE TABLE dept_salary_stats
SELECT
    dept_id,
    COUNT(*)          AS emp_count,
    AVG(salary)       AS avg_salary,
    MAX(salary)       AS max_salary,
    MIN(salary)       AS min_salary,
    SUM(salary)       AS total_salary
FROM employees
GROUP BY dept_id;

-- 查看统计结果
SELECT * FROM dept_salary_stats ORDER BY avg_salary
DESC;
```

2.6 FROM 子句多目标插入

```sql
-- 创建男女员工分别存放的表
CREATE TABLE IF NOT EXISTS male_employees (
    emp_id    STRING,
    emp_name  STRING,
    age       INT,
    salary    DOUBLE,
    dept_id   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

CREATE TABLE IF NOT EXISTS female_employees (
    emp_id    STRING,
    emp_name  STRING,
    age       INT,
    salary    DOUBLE,
    dept_id   STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

```
-- 한 번의 스캔으로 두 테이블에 동시 삽입
FROM employees
INSERT INTO TABLE male_employees
    SELECT emp_id, emp_name, age, salary, dept_id
WHERE gender = '男'
INSERT INTO TABLE female_employees
    SELECT emp_id, emp_name, age, salary, dept_id
WHERE gender = '女';

-- 결과 확인
SELECT '남직원' AS type, COUNT(*) AS cnt FROM
male_employees
UNION ALL
SELECT '여직원', COUNT(*) FROM female_employees;
```

🖊 제 2 장 연습문제

연습 2-1: `emp_test` 테이블에 자신의 정보를 직접 삽입하세요 (직원 번호는 `T999`, 이름은 영문 표기, 월급은 자유 설정). 삽입 후 결과를 조회하여 확인하세요.

연습 2-2: `shanghai_employees`라는 테이블을 생성하고, `employees` 테이블에서 상하이 (上海)에 근무하는 직원의 정보 (`emp_id`, `emp_name`, `salary`, `hire_date`) 를 삽입한 뒤 몇 명인지 집계하세요.

연습 2-3: `sales_summary` 테이블을 생성하고, `sales` 테이블을 지역 (`region`) 별로 집계하여 총 판매액·평균 판매액·최대 단건 판매액을 `INSERT OVERWRITE` 방식으로 저장하세요.

연습 2-4 (도전 문제) : `FROM` 다중 대상 삽입을 활용하여, `orders` 테이블에서 `已完成` (완료) 주문과 `已取消` (취소) 주문을 각각 `completed_orders`와 `cancelled_orders` 테이블에

```
-- 一次扫描，同时向两张表插入数据
FROM employees
INSERT INTO TABLE male_employees
    SELECT emp_id, emp_name, age, salary, dept_id
WHERE gender = '男'
INSERT INTO TABLE female_employees
    SELECT emp_id, emp_name, age, salary, dept_id
WHERE gender = '女';

-- 验证结果
SELECT '男员工' AS type, COUNT(*) AS cnt FROM
male_employees
UNION ALL
SELECT '女员工', COUNT(*) FROM female_employees;
```

🖊 第 2 章 练习题

练习 2-1：向 `emp_test` 表中手动插入你自己的信息（假设员工编号为 `T999`，姓名用拼音，月薪自定义），并查询验证。

练习 2-2：创建一个名为 `shanghai_employees` 的表，将 `employees` 表中所有在上海工作的员工信息（`emp_id`, `emp_name`, `salary`, `hire_date`）插入其中，并统计有多少名员工。

练习 2-3：创建一张 `sales_summary` 表，将 `sales` 表按 `region`（区域）统计各区域的总销售额、平均销售额、最大单笔销售额，并用 `INSERT OVERWRITE` 方式存入该表。

练习 2-4（挑战题）：利用 `FROM` 多目标插入，将 `orders` 表中 `已完成` 和 `已取消` 的订单分别插入 `completed_orders` 和 `cancelled_orders` 两张表中。

삽입하세요.

🔍 제 3 장 SELECT 기본 조회

학습 목표

- SELECT 문의 완전한 문법 습득
- WHERE 조건 필터링 사용법 습득
- ORDER BY 정렬 습득
- LIMIT 페이징 이해

3.1 SELECT 기본 문법

```sql
SELECT [DISTINCT] 컬럼1, 컬럼2, ...
FROM 테이블명
[WHERE 조건]
[GROUP BY 컬럼1, 컬럼2, ...]
[HAVING 조건]
[ORDER BY 컬럼1 [ASC|DESC], ...]
[LIMIT n];
```

3.2 기본 조회 예제

```sql
USE unit5db;

-- ① 전체 컬럼 조회
SELECT * FROM employees;

-- ② 특정 컬럼 조회
SELECT emp_id, emp_name, salary FROM employees;

-- ③ 컬럼 별칭 사용
SELECT
    emp_id   AS 직원번호,
    emp_name AS 이름,
    salary   AS 월급
FROM employees;
```

🔍 第 3 章 SELECT 基础查询

学习目标

- 掌握 SELECT 语句的完整语法
- 学会使用 WHERE 条件过滤
- 掌握 ORDER BY 排序
- 了解 LIMIT 分页

3.1 SELECT 基本语法

```sql
SELECT [DISTINCT] col1, col2, ...
FROM tablename
[WHERE condition]
[GROUP BY col1, col2, ...]
[HAVING condition]
[ORDER BY col1 [ASC|DESC], ...]
[LIMIT n];
```

3.2 基础查询示例

```sql
USE unit5db;

-- ① 查询所有列
SELECT * FROM employees;

-- ② 查询指定列
SELECT emp_id, emp_name, salary FROM employees;

-- ③ 列别名
SELECT
    emp_id   AS 员工编号,
    emp_name AS 姓名,
    salary   AS 月薪
FROM employees;
```

```sql
-- ④ 중복 제거 조회
SELECT DISTINCT city FROM employees;

-- ⑤ 계산 컬럼 (월급을 연봉으로 환산)
SELECT
    emp_name,
    salary,
    salary * 12 AS 연봉
FROM employees;
```

3.3 WHERE 조건 조회

```sql
-- ① 등호 조건
SELECT * FROM employees WHERE gender = '女';

-- ② 비교 연산자
SELECT emp_name, salary FROM employees WHERE salary
> 14000;

-- ③ 범위 조회 (BETWEEN)
SELECT emp_name, age FROM employees WHERE age
BETWEEN 28 AND 33;

-- ④ IN 조회
SELECT emp_name, city FROM employees WHERE city IN
('北京', '上海');

-- ⑤ LIKE 와일드카드 조회 (张씨 성 직원)
SELECT * FROM employees WHERE emp_name LIKE '张%';

-- ⑥ IS NULL / IS NOT NULL
SELECT * FROM employees WHERE dept_id IS NOT NULL;

-- ⑦ 다중 조건 조합 (AND / OR)
SELECT emp_name, gender, salary
FROM employees
WHERE gender = '女' AND salary >= 12000;

SELECT emp_name, city, salary
FROM employees
WHERE city = '北京' OR city = '上海';
```

3.4 ORDER BY 정렬

```sql
```


```sql
-- ④ 去重查询
SELECT DISTINCT city FROM employees;

-- ⑤ 计算列（薪资折算为年薪）
SELECT
    emp_name,
    salary,
    salary * 12 AS annual_salary
FROM employees;
```

3.3 WHERE 条件查询

```sql
-- ① 等于条件
SELECT * FROM employees WHERE gender = '女';

-- ② 比较运算符
SELECT emp_name, salary FROM employees WHERE salary
> 14000;

-- ③ 范围查询（BETWEEN）
SELECT emp_name, age FROM employees WHERE age
BETWEEN 28 AND 33;

-- ④ IN 查询
SELECT emp_name, city FROM employees WHERE city IN
('北京', '上海');

-- ⑤ LIKE 模糊查询（姓张的员工）
SELECT * FROM employees WHERE emp_name LIKE '张%';

-- ⑥ IS NULL / IS NOT NULL
SELECT * FROM employees WHERE dept_id IS NOT NULL;

-- ⑦ 多条件组合（AND / OR）
SELECT emp_name, gender, salary
FROM employees
WHERE gender = '女' AND salary >= 12000;

SELECT emp_name, city, salary
FROM employees
WHERE city = '北京' OR city = '上海';
```

3.4 ORDER BY 排序

```sql
```

```sql
-- ① 월급 내림차순 정렬
SELECT emp_name, salary FROM employees ORDER BY
salary DESC;

-- ② 도시 오름차순, 월급 내림차순 (다중 컬럼 정렬)
SELECT emp_name, city, salary
FROM employees
ORDER BY city ASC, salary DESC;

-- ③ 월급 상위 3 명 조회
SELECT emp_name, salary, dept_id
FROM employees
ORDER BY salary DESC
LIMIT 3;

-- ④ 가장 최근에 입사한 직원 5 명 조회
SELECT emp_name, hire_date, dept_id
FROM employees
ORDER BY hire_date DESC
LIMIT 5;
```

3.5 CASE WHEN 조건 표현식

```sql
-- 월급 범위에 따라 직원에게 등급 레이블 부여
SELECT
    emp_name,
    salary,
    CASE
        WHEN salary >= 18000 THEN '고임금'
        WHEN salary >= 12000 THEN '중간임금'
        ELSE '기본급'
    END AS 월급등급
FROM employees
ORDER BY salary DESC;
```

🖉 제 3 장 연습문제

연습 3-1: 나이가 30 세 이상 (30 세 포함) 인 남성 직원을 조회하고, 이름·나이·소재 도시·월급을 표시하며 월급 내림차순으로 정렬하세요.

연습 3-2: `sales` 테이블에서 2023 년 2 월의 모든 판매 기록 (`sale_date LIKE '2023-02%'`) 을

```sql
-- ① 按薪资降序排列
SELECT emp_name, salary FROM employees ORDER BY
salary DESC;

-- ② 按城市升序、薪资降序 (多列排序)
SELECT emp_name, city, salary
FROM employees
ORDER BY city ASC, salary DESC;

-- ③ 查询薪资最高的 3 名员工
SELECT emp_name, salary, dept_id
FROM employees
ORDER BY salary DESC
LIMIT 3;

-- ④ 查询最近入职的 5 名员工
SELECT emp_name, hire_date, dept_id
FROM employees
ORDER BY hire_date DESC
LIMIT 5;
```

3.5 CASE WHEN 条件表达式

```sql
-- 按薪资范围给员工打标签
SELECT
    emp_name,
    salary,
    CASE
        WHEN salary >= 18000 THEN '高薪'
        WHEN salary >= 12000 THEN '中薪'
        ELSE '基础薪资'
    END AS salary_level
FROM employees
ORDER BY salary DESC;
```

🖉 第 3 章 练习题

练习 3-1：查询所有年龄在 30 岁以上（含 30 岁）的男性员工，显示姓名、年龄、所在城市和月薪，按月薪降序排列。

练习 3-2：查询 `sales` 表中 2023 年 2 月份的所有销售记录（`sale_date LIKE '2023-02%'`），显示

조회하고, 판매 번호·직원 번호·판매 금액·제품 유형을 표시하며 금액 내림차순으로 정렬하세요.

연습 3-3: `orders` 테이블에서 상태가 `已完成`（완료）인 주문을 조회하고, 주문 번호·고객 번호·주문 날짜·총금액을 표시하되 총금액이 100,000 원을 초과하는 레코드만 표시하세요.

연습 3-4: CASE WHEN 을 사용하여 `employees`의 직원 나이를 구간별로 표시하세요: 25 세 미만은 `신세대`, 25~35 세는 `핵심인력`, 35 세 초과는 `시니어직원`. 이름·나이·구간 레이블을 표시하고 나이 오름차순으로 정렬하세요.

📊 제 4 장 집계 조회 및 그룹 통계

학습 목표

- 집계 함수 습득: COUNT, SUM, AVG, MAX, MIN
- GROUP BY 그룹 통계 습득
- HAVING 으로 그룹 결과 필터링 습득

4.1 자주 쓰는 집계 함수

```sql
USE unit5db;

-- ① COUNT: 행 수 집계
SELECT COUNT(*) AS total_employees FROM employees;
SELECT COUNT(DISTINCT city) AS city_count FROM employees;

-- ② SUM: 합계
SELECT SUM(salary) AS total_salary FROM employees;

-- ③ AVG: 평균
SELECT ROUND(AVG(salary), 2) AS avg_salary FROM employees;
```

销售编号、员工编号、销售金额、产品类型，按金额降序排列。

练习 3-3：查询 `orders` 表中状态为"已完成"的订单，显示订单编号、客户编号、订单日期、总金额，并只显示总金额超过 100000 元的记录。

练习 3-4：使用 CASE WHEN，对 `employees` 中的员工年龄分段标记：25 岁以下为"新生代"，25-35 岁为"中坚力量"，35 岁以上为"资深员工"。显示姓名、年龄和分段标签，按年龄升序排列。

📊 第 4 章 聚合查询与分组统计

学习目标

- 掌握聚合函数：COUNT、SUM、AVG、MAX、MIN
- 掌握 GROUP BY 分组统计
- 掌握 HAVING 对分组结果过滤

4.1 常用聚合函数

```sql
USE unit5db;

-- ① COUNT：统计行数
SELECT COUNT(*) AS total_employees FROM employees;
SELECT COUNT(DISTINCT city) AS city_count FROM employees;

-- ② SUM：求和
SELECT SUM(salary) AS total_salary FROM employees;

-- ③ AVG：平均值
SELECT ROUND(AVG(salary), 2) AS avg_salary FROM employees;
```

```sql
-- ④ MAX / MIN: 최대값 / 최소값
SELECT MAX(salary) AS max_salary, MIN(salary) AS
min_salary FROM employees;

-- ⑤ 복합 사용
SELECT
    COUNT(*)              AS 총직원수,
    ROUND(AVG(salary), 0) AS 평균월급,
    MAX(salary)           AS 최고월급,
    MIN(salary)           AS 최저월급,
    SUM(salary)           AS 월급합계
FROM employees;
```

4.2 GROUP BY 그룹 통계

```sql
-- ① 부서별 직원 수 집계
SELECT dept_id, COUNT(*) AS emp_count
FROM employees
GROUP BY dept_id
ORDER BY emp_count DESC;

-- ② 부서별 평균 월급 집계
SELECT
    dept_id,
    COUNT(*)              AS emp_count,
    ROUND(AVG(salary), 0) AS avg_salary
FROM employees
GROUP BY dept_id
ORDER BY avg_salary DESC;

-- ③ 도시별 직원 분포 집계
SELECT city, COUNT(*) AS emp_count
FROM employees
GROUP BY city
ORDER BY emp_count DESC;

-- ④ 성별 평균 월급 집계
SELECT
    gender,
    COUNT(*) AS emp_count,
    ROUND(AVG(salary), 0) AS avg_salary
FROM employees
GROUP BY gender;

-- ⑤ 제품 유형별 판매 현황 집계
SELECT
    product_type,
    COUNT(*)              AS sale_count,
    SUM(amount)           AS total_amount,
    ROUND(AVG(amount), 0) AS avg_amount
FROM sales
GROUP BY product_type
```


```sql
-- ④ MAX / MIN : 最大值/最小值
SELECT MAX(salary) AS max_salary, MIN(salary) AS
min_salary FROM employees;

-- ⑤ 综合使用
SELECT
    COUNT(*)              AS 员工总数,
    ROUND(AVG(salary), 0) AS 平均月薪,
    MAX(salary)           AS 最高月薪,
    MIN(salary)           AS 最低月薪,
    SUM(salary)           AS 月薪总额
FROM employees;
```

4.2 GROUP BY 分组统计

```sql
-- ① 按部门统计员工人数
SELECT dept_id, COUNT(*) AS emp_count
FROM employees
GROUP BY dept_id
ORDER BY emp_count DESC;

-- ② 按部门统计平均薪资
SELECT
    dept_id,
    COUNT(*)              AS emp_count,
    ROUND(AVG(salary), 0) AS avg_salary
FROM employees
GROUP BY dept_id
ORDER BY avg_salary DESC;

-- ③ 按城市统计员工分布
SELECT city, COUNT(*) AS emp_count
FROM employees
GROUP BY city
ORDER BY emp_count DESC;

-- ④ 按性别统计平均薪资
SELECT
    gender,
    COUNT(*) AS emp_count,
    ROUND(AVG(salary), 0) AS avg_salary
FROM employees
GROUP BY gender;

-- ⑤ 按产品类型统计销售情况
SELECT
    product_type,
    COUNT(*)          AS sale_count,
    SUM(amount)       AS total_amount,
    ROUND(AVG(amount), 0) AS avg_amount
FROM sales
GROUP BY product_type
```

```
ORDER BY total_amount DESC;
```

4.3 HAVING 그룹 결과 필터링

> 💡 **WHERE vs HAVING**:
> - `WHERE`: 그룹화 **전** 행 필터링
> - `HAVING`: 그룹화 **후** 그룹 필터링

```sql
-- ① 평균 월급이 14,000 원을 초과하는 부서만 표시
SELECT
    dept_id,
    COUNT(*) AS emp_count,
    ROUND(AVG(salary), 0) AS avg_salary
FROM employees
GROUP BY dept_id
HAVING AVG(salary) > 14000
ORDER BY avg_salary DESC;

-- ② 총 판매액이 200,000 원을 초과하는 지역만 표시
SELECT
    region,
    COUNT(*)    AS sale_count,
    SUM(amount) AS total_amount
FROM sales
GROUP BY region
HAVING SUM(amount) > 200000
ORDER BY total_amount DESC;

-- ③ WHERE + GROUP BY + HAVING 복합 사용
-- 2023 년 2 월 이후 판매액 합계가 100,000 원을 초과하는 직원 집계
SELECT
    emp_id,
    COUNT(*)    AS sale_count,
    SUM(amount) AS total_amount
FROM sales
WHERE sale_date >= '2023-02-01'
GROUP BY emp_id
HAVING SUM(amount) > 100000
ORDER BY total_amount DESC;
```

4.4 종합 통계 사례

```sql
-- 사례: 지역별·제품 유형별 판매 현황 집계
SELECT
    region,
```


```
ORDER BY total_amount DESC;
```

4.3 HAVING 过滤分组结果

> 💡 **WHERE vs HAVING** ：
> - `WHERE`：在分组**前**过滤行
> - `HAVING`：在分组**后**过滤组

```sql
-- ① 只显示平均薪资超过 14000 的部门
SELECT
    dept_id,
    COUNT(*) AS emp_count,
    ROUND(AVG(salary), 0) AS avg_salary
FROM employees
GROUP BY dept_id
HAVING AVG(salary) > 14000
ORDER BY avg_salary DESC;

-- ② 只显示销售额超过 200000 的区域
SELECT
    region,
    COUNT(*)    AS sale_count,
    SUM(amount) AS total_amount
FROM sales
GROUP BY region
HAVING SUM(amount) > 200000
ORDER BY total_amount DESC;

-- ③ WHERE + GROUP BY + HAVING 综合使用
-- 统计 2023 年 2 月以后，销售额合计超过 100000 的员工
SELECT
    emp_id,
    COUNT(*)    AS sale_count,
    SUM(amount) AS total_amount
FROM sales
WHERE sale_date >= '2023-02-01'
GROUP BY emp_id
HAVING SUM(amount) > 100000
ORDER BY total_amount DESC;
```

4.4 综合统计案例

```sql
-- 案例：统计各区域各产品类型的销售情况
SELECT
    region,
```

```
    product_type,
    COUNT(*)              AS sale_times,
    SUM(amount)           AS total_sales,
    ROUND(AVG(amount), 0) AS avg_sales,
    MAX(amount)           AS max_single_sale
FROM sales
GROUP BY region, product_type
ORDER BY region, total_sales DESC;
```

🖊 제 4 장 연습문제

연습 4-1: `employees` 테이블에서 각 부서（`dept_id`）의 성별별 직원 수와 평균 월급을 집계하고, 부서 번호와 성별 순으로 정렬하세요.

연습 4-2: `sales` 테이블에서 각 직원（`emp_id`）의 총 판매액을 집계하고, 총 판매액이 200,000 원을 초과하는 직원만 표시하며 총 판매액 내림차순으로 정렬하세요.

연습 4-3: `orders` 테이블에서 각 상태（`status`）별 주문 수량과 총금액을 집계하고, `HAVING`을 사용하여 주문 수량이 5 건을 초과하는 상태만 표시하세요.

연습 4-4（도전 문제）: `employees` 테이블에서 도시별 직원 정보를 집계하세요. 조건: 도시명·직원 수·평균 월급·최고 월급을 표시하고, 직원 수가 2 명을 초과하는 도시만 표시하며 평균 월급 내림차순으로 정렬하세요.

🔗 제 5 장 다중 테이블 조인 조회（JOIN）

학습 목표

- JOIN 의 개념과 종류 이해

```
    product_type,
    COUNT(*)          AS sale_times,
    SUM(amount)       AS total_sales,
    ROUND(AVG(amount), 0) AS avg_sales,
    MAX(amount)       AS max_single_sale
FROM sales
GROUP BY region, product_type
ORDER BY region, total_sales DESC;
```

🖊 第 4 章 练习题

练习 4-1：统计 `employees` 表中各部门（`dept_id`）男女员工的人数和平均薪资，结果按部门编号和性别排序。

练习 4-2：统计 `sales` 表中各员工（`emp_id`）的总销售额，只显示总销售额超过 200000 元的员工，按总销售额降序排列。

练习 4-3：统计 `orders` 表中各状态（`status`）的订单数量和总金额，并用 `HAVING` 只保留订单数量大于 5 的状态。

练习 4-4（挑战题）：统计 `employees` 表中各城市的员工信息，要求：显示城市名、员工人数、平均薪资、最高薪资；只显示员工人数超过 2 人的城市；按平均薪资降序排列。

🔗 第 5 章 多表连接查询（JOIN）

学习目标

- 理解 JOIN 的概念和类型

- INNER JOIN, LEFT JOIN, RIGHT JOIN 습득
- 다중 테이블 연관 조회 습득

5.1 JOIN 유형 설명

JOIN 유형	설명
`INNER JOIN`	두 테이블 모두에서 조건이 일치하는 레코드만 반환
`LEFT JOIN`	왼쪽 테이블의 모든 레코드 반환, 오른쪽 테이블에 일치 항목 없으면 NULL
`RIGHT JOIN`	오른쪽 테이블의 모든 레코드 반환, 왼쪽 테이블에 일치 항목 없으면 NULL
`FULL OUTER JOIN`	두 테이블의 모든 레코드 반환, 일치 항목 없으면 NULL 로 채움

5.2 INNER JOIN（내부 조인）

```sql
USE unit5db;

-- ① 직원 테이블과 부서 테이블을 연결하여 직원 이름과 소재 부서명 표시
SELECT
    e.emp_id,
    e.emp_name,
    e.salary,
    d.dept_name,
    d.location
FROM employees e
INNER JOIN departments d ON e.dept_id = d.dept_id
ORDER BY e.salary DESC;

-- ② 화동（华东）지역 판매 직원 정보 조회
SELECT
    e.emp_name,
    e.city,
    s.sale_date,
    s.amount,
    s.product_type
FROM sales s
INNER JOIN employees e ON s.emp_id = e.emp_id
WHERE s.region = '华东'
```

- 掌握 INNER JOIN、LEFT JOIN、RIGHT JOIN
- 学会多表关联查询

5.1 JOIN 类型说明

JOIN 类型	说明
`INNER JOIN`	只返回两表都有匹配的记录
`LEFT JOIN`	返回左表所有记录，右表无匹配则为 NULL
`RIGHT JOIN`	返回右表所有记录，左表无匹配则为 NULL
`FULL OUTER JOIN`	返回两表所有记录，无匹配的用 NULL 填充

5.2 INNER JOIN（内连接）

```sql
USE unit5db;

-- ① 员工表与部门表关联，显示员工姓名和所在部门名称

SELECT
    e.emp_id,
    e.emp_name,
    e.salary,
    d.dept_name,
    d.location
FROM employees e
INNER JOIN departments d ON e.dept_id = d.dept_id
ORDER BY e.salary DESC;

-- ② 查询华东区域的销售员工信息
SELECT
    e.emp_name,
    e.city,
    s.sale_date,
    s.amount,
    s.product_type
FROM sales s
INNER JOIN employees e ON s.emp_id = e.emp_id
WHERE s.region = '华东'
```

```
ORDER BY s.amount DESC;
```

5.3 LEFT JOIN (왼쪽 조인)

```sql
-- 직원이 없는 부서도 포함하여 모든 부서의 직원 수 표시
SELECT
    d.dept_id,
    d.dept_name,
    COUNT(e.emp_id) AS emp_count
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_id, d.dept_name
ORDER BY emp_count DESC;

-- 주문이 없는 직원도 포함하여 모든 직원의 주문 건수 표시
SELECT
    e.emp_id,
    e.emp_name,
    e.dept_id,
    COUNT(o.order_id) AS order_count
FROM employees e
LEFT JOIN orders o ON e.emp_id = o.emp_id
GROUP BY e.emp_id, e.emp_name, e.dept_id
ORDER BY order_count DESC;
```

5.4 다중 테이블 조인 조회

```sql
-- 3 테이블 조인: 주문 + 직원 + 부서
SELECT
    o.order_id,
    o.customer_id,
    o.total_amount,
    o.status,
    e.emp_name,
    d.dept_name
FROM orders o
INNER JOIN employees e ON o.emp_id = e.emp_id
INNER JOIN departments d ON e.dept_id = d.dept_id
ORDER BY o.total_amount DESC
LIMIT 10;

-- 4 테이블 조인: 주문 상세 + 주문 + 직원 + 부서
SELECT
    oi.item_id,
    oi.product_name,
    oi.quantity,
    oi.unit_price,
    oi.quantity * oi.unit_price AS subtotal,
```


```
ORDER BY s.amount DESC;
```

5.3 LEFT JOIN（左连接）

```sql
-- 显示所有部门及其员工数量（包括没有员工的部门）
SELECT
    d.dept_id,
    d.dept_name,
    COUNT(e.emp_id) AS emp_count
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_id, d.dept_name
ORDER BY emp_count DESC;

-- 显示所有员工及其订单（包括没有订单的员工）

SELECT
    e.emp_id,
    e.emp_name,
    e.dept_id,
    COUNT(o.order_id) AS order_count
FROM employees e
LEFT JOIN orders o ON e.emp_id = o.emp_id
GROUP BY e.emp_id, e.emp_name, e.dept_id
ORDER BY order_count DESC;
```

5.4 多表连接查询

```sql
-- 三表关联：订单 + 员工 + 部门
SELECT
    o.order_id,
    o.customer_id,
    o.total_amount,
    o.status,
    e.emp_name,
    d.dept_name
FROM orders o
INNER JOIN employees e ON o.emp_id = e.emp_id
INNER JOIN departments d ON e.dept_id = d.dept_id
ORDER BY o.total_amount DESC
LIMIT 10;

-- 四表关联：订单明细 + 订单 + 员工 + 部门
SELECT
    oi.item_id,
    oi.product_name,
    oi.quantity,
    oi.unit_price,
    oi.quantity * oi.unit_price AS subtotal,
```

```
    o.order_id,
    o.status,
    e.emp_name
FROM order_items oi
INNER JOIN orders o      ON oi.order_id =
o.order_id
INNER JOIN employees e   ON o.emp_id = e.emp_id
WHERE o.status = '已完成'
ORDER BY subtotal DESC
LIMIT 10;
```

5.5 종합 JOIN 사례

```sql
-- 부서별 판매 실적 종합 보고서
SELECT
    d.dept_name   AS 부서명,
    d.location    AS 소재도시,
    COUNT(DISTINCT e.emp_id) AS 직원수,
    COUNT(s.sale_id)         AS 판매횟수,
    SUM(s.amount)            AS 총판매액,
    ROUND(AVG(s.amount), 0)  AS 평균단건판매액
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
LEFT JOIN sales s     ON e.emp_id = s.emp_id
GROUP BY d.dept_name, d.location
ORDER BY 총판매액 DESC;
```

🖊 제 5 장 연습문제

연습 5-1: `INNER JOIN`으로 `employees`와 `departments`를 연결하여 각 직원의 이름·부서명·도시·월급을 표시하고, 월급 내림차순으로 정렬하여 상위 10 명만 표시하세요.

연습 5-2: `LEFT JOIN`으로 모든 직원과 판매 기록을 조회하여 직원 이름·판매 날짜·판매 금액을 표시하세요 (판매 기록이 없는 직원은 금액을 0 으로 표시). 직원별 판매 횟수도 집계하세요.


```
    o.order_id,
    o.status,
    e.emp_name
FROM order_items oi
INNER JOIN orders o      ON oi.order_id =
o.order_id
INNER JOIN employees e   ON o.emp_id = e.emp_id
WHERE o.status = '已完成'
ORDER BY subtotal DESC
LIMIT 10;
```

5.5 综合 JOIN 案例

```sql
-- 各部门销售业绩汇总报表
SELECT
    d.dept_name   AS 部门名称,
    d.location    AS 所在城市,
    COUNT(DISTINCT e.emp_id) AS 员工人数,
    COUNT(s.sale_id)         AS 销售次数,
    SUM(s.amount)            AS 总销售额,
    ROUND(AVG(s.amount), 0)  AS 平均单笔销售额
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
LEFT JOIN sales s     ON e.emp_id = s.emp_id
GROUP BY d.dept_name, d.location
ORDER BY 总销售额 DESC;
```

🖊 第 5 章 练习题

练习 5-1：使用 `INNER JOIN` 连接 `employees` 和 `departments`，显示每位员工的姓名、所属部门名称、城市和月薪，结果按月薪降序排列，只显示前 10 名。

练习 5-2：使用 `LEFT JOIN` 查询所有员工及其销售记录，显示员工姓名、销售日期、销售金额（如果该员工没有销售记录，金额显示为 0），统计每个员工的销售次数。

연습 5-3: `departments`·`employees`·`sales` 3 개 테이블을 조인하여 각 부서별 제품 유형별 총 판매액을 집계하고, 부서명과 총 판매액 기준으로 정렬하세요.

연습 5-4（도전 문제）: 각 고객（`customer_id`）의 총 주문금액과 주문 건수, 그리고 해당 고객의 주문을 가장 많이 담당한 직원 이름을 조회하세요（`orders`와 `employees` 조인 필요）.

🪆 제 6 장 서브쿼리 및 중첩 쿼리

학습 목표

- WHERE 절의 서브쿼리 습득
- IN, EXISTS 사용 이해
- FROM 절의 서브쿼리（파생 테이블）활용

6.1 WHERE 절의 서브쿼리

```sql
USE unit5db;

-- ① 평균 월급보다 높은 직원 조회
SELECT emp_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)
ORDER BY salary DESC;

-- ② 왕치앙（王强）과 같은 부서의 직원 조회
SELECT emp_name, dept_id, salary
FROM employees
WHERE dept_id = (
    SELECT dept_id FROM employees WHERE emp_name =
'王强'
)
AND emp_name != '王强';

-- ③ IN 서브쿼리: 판매 기록이 있는 직원 조회
SELECT emp_id, emp_name, dept_id, salary
```

练习 5-3：编写一个查询，统计每个部门下各产品类型的销售总额。需要连接 `departments`、`employees`、`sales` 三张表，结果按部门名称和总销售额排序。

练习 5-4（挑战题）：查询每个客户（`customer_id`）的订单总金额和订单数量，以及负责该客户订单最多的员工姓名（需要连接 `orders` 和 `employees` 表）。

🪆 第 6 章 子查询与嵌套查询

学习目标

- 掌握 WHERE 子句中的子查询
- 理解 IN、EXISTS 的使用
- 学会在 FROM 子句中使用子查询

6.1 WHERE 中的子查询

```sql
USE unit5db;

-- ① 查询薪资高于平均薪资的员工
SELECT emp_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)
ORDER BY salary DESC;

-- ② 查询与"王强"在同一部门的员工
SELECT emp_name, dept_id, salary
FROM employees
WHERE dept_id = (
    SELECT dept_id FROM employees WHERE emp_name =
'王强'
)
AND emp_name != '王强';

-- ③ IN 子查询：查询有销售记录的员工信息
SELECT emp_id, emp_name, dept_id, salary
```

```
FROM employees
WHERE emp_id IN (
    SELECT DISTINCT emp_id FROM sales
);

-- ④ NOT IN 서브쿼리: 판매 기록이 없는 직원 조회
SELECT emp_id, emp_name, dept_id, salary
FROM employees
WHERE emp_id NOT IN (
    SELECT DISTINCT emp_id FROM sales
);
```

6.2 FROM 절의 서브쿼리 (파생 테이블)

```sql
-- 서브쿼리 결과를 임시 테이블로 사용
-- 부서별 평균 월급을 조회한 후 전사 평균 이상인 부서만 표시
SELECT dept_id, avg_sal
FROM (
    SELECT dept_id, ROUND(AVG(salary), 0) AS avg_sal
    FROM employees
    GROUP BY dept_id
) t
WHERE avg_sal > (SELECT AVG(salary) FROM employees)
ORDER BY avg_sal DESC;

-- 판매 실적 상위 3 명의 직원 상세 정보 조회
SELECT e.emp_name, e.dept_id, e.salary,
s.total_sales
FROM employees e
INNER JOIN (
    SELECT emp_id, SUM(amount) AS total_sales
    FROM sales
    GROUP BY emp_id
    ORDER BY total_sales DESC
    LIMIT 3
) s ON e.emp_id = s.emp_id
ORDER BY s.total_sales DESC;
```

🖉 제 6 장 연습문제

연습 6-1: 서브쿼리를 사용하여 월급이 가장 높은 직원이 속한 부서의 모든 직원 정보를 조회하세요 (직원 이름이나 부서 번호를 직접 조건으로 사용하면 안 됩니다).

```
FROM employees
WHERE emp_id IN (
    SELECT DISTINCT emp_id FROM sales
);

-- ④ NOT IN 子查询：查询没有销售记录的员工
SELECT emp_id, emp_name, dept_id, salary
FROM employees
WHERE emp_id NOT IN (
    SELECT DISTINCT emp_id FROM sales
);
```

6.2 FROM 子句中的子查询（派生表）

```sql
-- 使用子查询结果作为临时表
-- 查询各部门平均薪资，然后只显示高于全公司平均的部门

SELECT dept_id, avg_sal
FROM (
    SELECT dept_id, ROUND(AVG(salary), 0) AS avg_sal
    FROM employees
    GROUP BY dept_id
) t
WHERE avg_sal > (SELECT AVG(salary) FROM employees)
ORDER BY avg_sal DESC;

-- 找出销售业绩排名前 3 的员工详情
SELECT e.emp_name, e.dept_id, e.salary,
s.total_sales
FROM employees e
INNER JOIN (
    SELECT emp_id, SUM(amount) AS total_sales
    FROM sales
    GROUP BY emp_id
    ORDER BY total_sales DESC
    LIMIT 3
) s ON e.emp_id = s.emp_id
ORDER BY s.total_sales DESC;
```

🖉 第 6 章 练习题

练习 6-1：使用子查询，找出薪资最高的员工所在部门的所有员工信息（不能直接使用员工姓名或部门编号作为条件）。

연습 6-2: `orders` 테이블에서 주문 총금액이 모든 주문의 평균 총금액보다 높은 주문을 조회하여 주문 번호·고객 번호·총금액·상태를 표시하세요.

연습 6-3: `NOT IN` 서브쿼리를 사용하여 `orders` 테이블에 주문이 없는 직원（`orders.emp_id`에 등장하지 않는 직원）을 조회하고, 직원 번호·이름·부서 번호를 표시하세요.

📥 제 7 장 데이터 내보내기 작업

학습 목표

- `INSERT OVERWRITE DIRECTORY`로 HDFS 내보내기 습득
- `INSERT OVERWRITE LOCAL DIRECTORY`로 로컬 내보내기 습득
- 내보내기 형식 설정 이해

7.1 HDFS 로 내보내기

```sql
USE unit5db;

-- 고임금 직원 데이터를 HDFS 로 내보내기
INSERT OVERWRITE DIRECTORY
'/user/hive/output/unit5/high_salary'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT emp_id, emp_name, salary, dept_id, city
FROM employees
WHERE salary >= 15000
ORDER BY salary DESC;
```

터미널에서 확인（새 터미널 열기）:

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

练习 6-2：查询 `orders` 表中，订单总金额高于所有订单平均总金额的订单信息，显示订单编号、客户编号、总金额、状态。

练习 6-3：使用 `NOT IN` 子查询，找出在 `orders` 表中没有接过订单的员工（即不在 `orders.emp_id` 中出现的员工），显示其编号、姓名、部门编号。

📥 第 7 章 数据导出操作

学习目标

- 掌握 `INSERT OVERWRITE DIRECTORY` 导出到 HDFS
- 掌握 `INSERT OVERWRITE LOCAL DIRECTORY` 导出到本地
- 了解导出格式设置

7.1 导出到 HDFS

```sql
USE unit5db;

-- 将高薪员工数据导出到 HDFS
INSERT OVERWRITE DIRECTORY
'/user/hive/output/unit5/high_salary'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT emp_id, emp_name, salary, dept_id, city
FROM employees
WHERE salary >= 15000
ORDER BY salary DESC;
```

在终端验证（打开新终端）：

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

7.2 로컬 파일 시스템으로 내보내기

```sql
-- 판매 통계 데이터를 로컬로 내보내기
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/sales_summary'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT
    region,
    product_type,
    COUNT(*)    AS sale_count,
    SUM(amount) AS total_amount
FROM sales
GROUP BY region, product_type
ORDER BY region, total_amount DESC;
```

터미널에서 확인:

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

7.3 탭 구분자로 내보내기 (Excel 용)

```sql
-- 탭 구분자로 내보내기 (Excel 에서 열기 편리)
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/dept_stats'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
SELECT
    d.dept_name,
    COUNT(e.emp_id)         AS emp_count,
    ROUND(AVG(e.salary), 0) AS avg_salary
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_name
ORDER BY avg_salary DESC;
```

🖊 제 7 장 연습문제

연습 7-1: `orders` 테이블에서 상태가 `已完成` (완료) 인 주문을 로컬 디렉터리

7.2 导出到本地文件系统

```sql
-- 将销售统计数据导出到本地
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/sales_summary'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT
    region,
    product_type,
    COUNT(*)    AS sale_count,
    SUM(amount) AS total_amount
FROM sales
GROUP BY region, product_type
ORDER BY region, total_amount DESC;
```

在终端验证：

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

7.3 导出为 JSON 格式数据

```sql
-- 导出为制表符分隔（便于 Excel 打开）
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/output/unit5/dept_stats'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
SELECT
    d.dept_name,
    COUNT(e.emp_id)         AS emp_count,
    ROUND(AVG(e.salary), 0) AS avg_salary
FROM departments d
LEFT JOIN employees e ON d.dept_id = e.dept_id
GROUP BY d.dept_name
ORDER BY avg_salary DESC;
```

🖊 第 7 章练习题

练习 7-1：将 `orders` 表中状态为"已完成"的订单导出到本地目录

`/home/hadoop/output/unit5/completed_orders/`로 내보내세요. 필드는 쉼표로 구분합니다.

연습 7-2: 부서별 직원 요약 통계（부서명·직원 수·총 월급）를 HDFS 경로 `/user/hive/output/unit5/dept_summary/`로 내보내세요.

🏆 제 8 장 종합 연습문제

> ⏱ 예상 시간: 40 분
> 이번 장에서 학습한 DML 지식을 종합적으로 활용하여 다음 도전 과제를 완성하세요.

종합 문제 1: 직원 성과 분석 보고서

과제 설명: `emp_performance_report`라는 종합 테이블을 생성하고 다음 필드를 포함하세요:

필드명	설명
`emp_id`	직원 번호
`emp_name`	직원 이름
`dept_name`	소속 부서명
`salary`	월급
`sale_count`	판매 횟수（판매 기록 없으면 0）
`total_sales`	총 판매액（판매 기록 없으면 0）
`order_count`	담당 주문 수（주문 없으면 0）
`salary_level`	월급 등급（고/중/저）

요구 사항:

`/home/hadoop/output/unit5/completed_orders/`，字段以逗号分隔。

练习 7-2：将各部门的员工信息汇总（部门名称、员工人数、总月薪）导出到 HDFS 路径 `/user/hive/output/unit5/dept_summary/`。

🏆 第 8 章 综合练习题

> ⏱ 预计时间：40 分钟
> 综合运用本章所学的 DML 操作知识完成以下挑战性练习题。

综合题 1：员工绩效分析报表

任务描述：创建一张名为 `emp_performance_report` 的汇总表，包含以下字段：

字段名	说明
`emp_id`	员工编号
`emp_name`	员工姓名
`dept_name`	所属部门名称
`salary`	月薪
`sale_count`	销售次数（无销售记录则为 0）
`total_sales`	总销售额（无销售记录则为 0）
`order_count`	负责订单数（无订单则为 0）
`salary_level`	薪资等级（高/中/低）

要求：

1. `CREATE TABLE`로 테이블 생성
2. `INSERT OVERWRITE ... SELECT`와 다중 테이블 JOIN 으로 데이터 채우기
3. 총 판매액 내림차순으로 정렬

종합 문제 2: 월별 판매 분석

과제 설명: 2023 년 각 월별 판매 현황을 집계하는 쿼리를 작성하세요.

표시 항목: 월별·판매 횟수·총 판매액·최고 단건 금액·판매 제품 종류 수·해당 월 판매 왕 이름（판매액 기준 1 위）.

> 힌트: `SUBSTR(sale_date, 1, 7)`로 연월 추출 가능

종합 문제 3: 데이터 정제 및 변환

과제 설명:
1. `employees` 테이블에서 월급이 소속 부서 평균 월급보다 낮은 직원을 모두 조회
2. 해당 직원 정보（직원 번호·이름·월급·부서 평균 월급·월급 차액）를 새 테이블 `below_avg_employees`에 저장
3. 해당 테이블 데이터를 로컬 파일 `/home/hadoop/output/unit5/below_avg/`로 내보내기

종합 문제 4: Top N 분석

1. 使用 `CREATE TABLE` 建表
2. 使用 `INSERT OVERWRITE ... SELECT` 结合多表 JOIN 填充数据
3. 结果按总销售额降序排列

综合题 2：月度销售分析

任务描述：编写一个查询，统计 2023 年各月份的销售情况：

要求显示：月份、销售次数、总销售额、最高单笔金额、销售的产品种类数、销售最好的员工姓名（销售额最高的那位）。

> 提示：可以使用 `SUBSTR(sale_date, 1, 7)` 提取年月部分

综合题 3：数据清洗与转换

任务描述：
1. 从 `employees` 表中找出所有薪资低于该部门平均薪资的员工
2. 将这些员工的信息（员工编号、姓名、薪资、部门平均薪资、薪资差额）存入新表 `below_avg_employees`
3. 导出该表数据到本地文件 `/home/hadoop/output/unit5/below_avg/`

综合题 4：Top N 分析

과제 설명: 각 지역（`region`）에서 판매액 상위 2 명의 판매 직원 정보를 조회하고, 지역명·직원 번호·직원 이름·소속 부서·해당 직원의 해당 지역 총 판매액을 표시하세요.

> 힌트: Hive 3.x 는 `ROW_NUMBER() OVER(PARTITION BY ... ORDER BY ...)` 윈도우 함수를 지원합니다

📌 실험 정리

DML 핵심 명령어 정리

작업 유형	핵심 명령어	사용 시나리오
데이터 로드	`LOAD DATA LOCAL INPATH`	로컬 파일 일괄 가져오기
추가 삽입	`INSERT INTO TABLE`	테이블에 데이터 추가
덮어쓰기 삽입	`INSERT OVERWRITE TABLE`	테이블의 모든 데이터 교체
쿼리 삽입	`INSERT INTO ... SELECT`	조회 결과로 데이터 생성
기본 조회	`SELECT ... WHERE ... ORDER BY`	데이터 검색 및 필터링
그룹 통계	`GROUP BY ... HAVING`	데이터 집계 분석
다중 테이블 연결	`JOIN ... ON`	테이블 간 데이터 통합
서브쿼리	`WHERE ... IN (SELECT ...)`	중첩 조건 조회
데이터 내보내기	`INSERT OVERWRITE DIRECTORY`	파일로 데이터 내보내기

任务描述：查询每个区域（`region`）中销售额最高的前 2 名销售员信息，显示区域名称、员工编号、员工姓名、所属部门、该员工在该区域的总销售额。

> 提示：Hive 3.x 支持 `ROW_NUMBER() OVER(PARTITION BY ... ORDER BY ...)` 窗口函数

📌 实验总结

DML 知识点回顾

操作类型	关键命令	使用场景
数据加载	`LOAD DATA LOCAL INPATH`	从本地文件批量导入
追加插入	`INSERT INTO TABLE`	向表中追加数据
覆盖插入	`INSERT OVERWRITE TABLE`	替换表中所有数据
查询插入	`INSERT INTO ... SELECT`	从查询结果创建数据
基础查询	`SELECT ... WHERE ... ORDER BY`	数据检索与筛选
分组统计	`GROUP BY ... HAVING`	数据聚合分析
多表关联	`JOIN ... ON`	跨表数据整合
子查询	`WHERE ... IN (SELECT ...)`	嵌套条件查询
数据导出	`INSERT OVERWRITE DIRECTORY`	数据导出到文件

자주 발생하는 오류와 해결 방법

오류 유형	가능한 원인	해결 방법
데이터 로드 후 행 수 두 배	OVERWRITE 미사용	`LOAD DATA LOCAL INPATH ... OVERWRITE INTO TABLE` 사용
JOIN 결과 행 수 이상	조인 조건 오류	`ON` 절의 필드가 올바른지 확인
GROUP BY 오류	SELECT 에 비집계·비 GROUP BY 컬럼 존재	SELECT 컬럼이 GROUP BY 에 있거나 집계 함수로 감싸야 함
HAVING 필터 미작동	집계 조건을 WHERE 에 작성	집계 함수 조건을 HAVING 절로 이동
내보낸 파일 비어 있음	쿼리 조건이 너무 엄격	SELECT 문이 데이터를 반환하는지 먼저 확인

워크북 끝

제출 안내: 실험 완료 후 핵심 단계의 실행 결과를 스크린샷으로 저장하고, 실험 보고서에 발생한 문제와 해결 과정을 기록하세요.

常见错误与解决方法

错误类型	可能原因	解决方法
数据加载后行数翻倍	没有使用 OVERWRITE	使用 `LOAD DATA LOCAL INPATH ... OVERWRITE INTO TABLE`
JOIN 结果行数异常	关联条件写错	检查 `ON` 子句中的字段是否正确
GROUP BY 报错	SELECT 中有非聚合、非 GROUP BY 的列	确保 SELECT 的列要么被 GROUP BY，要么用聚合函数包裹
HAVING 过滤无效	把聚合条件写在 WHERE 中	将聚合函数条件移到 HAVING 子句
导出文件为空	查询条件过于严格	先检查 SELECT 语句是否有返回数据

工作手册结束

提交说明：完成实验后，请截图保存关键步骤的执行结果，并在实验报告中记录遇到的问题和解决过程。