Hive 함수 실전 연습 워크북

프로젝트 7 Hive 함수 종합 실습 매뉴얼

실습 시간: 4 시간
데이터베이스 명: unit7db
사용 도구: Beeline (별칭: hive-bee)
데이터 파일 위치: /usr/datas/

실습 준비

단계 1: 데이터 파일 디렉토리 생성

```bash
# 1. 데이터 파일 저장 디렉토리 생성
sudo mkdir -p /usr/datas
sudo chmod 777 /usr/datas

# 2. 모든 데이터 파일을 /usr/datas 디렉토리에 다운로드
# (데이터 파일은 본 문서 말미에서 제공)
```

단계 2: Hive 연결 및 데이터베이스 생성

```bash
# beeline 으로 Hive 접속
hive-bee
```

```sql
-- 실습 데이터베이스 생성
CREATE DATABASE IF NOT EXISTS unit7db;

-- 데이터베이스 사용
USE unit7db;

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

제 1 부: 문자열 함수 실습 (40 분)

Hive 函数实战练习册

项目 7 Hive 函数综合实习手册

实习时间: 4 小时
数据库名称: unit7db
使用工具: Beeline (别名: hive-bee)
数据文件位置: /usr/datas/

实习准备

步骤 1: 创建数据文件目录

```bash
# 1. 创建数据文件存放目录
sudo mkdir -p /usr/datas
sudo chmod 777 /usr/datas

# 2. 下载所有数据文件到 /usr/datas 目录

# (数据文件将在本文档末尾提供下载)
```

步骤 2: 连接 Hive 并创建数据库

```bash
# 启动 beeline 连接 Hive
hive-bee
```

```sql
-- 创建实习数据库
CREATE DATABASE IF NOT EXISTS unit7db;

-- 使用数据库
USE unit7db;

-- 验证当前数据库
SELECT current_database();
```

第一部分: 字符串函数实习 (40 分钟)

연습 1.1: 직원 정보 테이블 생성

```sql
-- 직원 테이블 생성
CREATE TABLE IF NOT EXISTS employees (
    emp_id INT,
    emp_name STRING,
    email STRING,
    phone STRING,
    department STRING,
    salary DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH '/usr/datas/employees.txt'
OVERWRITE INTO TABLE employees;

-- 데이터 확인
SELECT * FROM employees;
SELECT count(*) as total_count FROM employees;
```

데이터 파일: employees.txt

연습 1.2: 문자열 길이와 대소문자 변환

```sql
-- 과제 1: 직원 이름의 길이 계산
SELECT
    emp_name,
    length(emp_name) AS name_length
FROM employees;

-- 과제 2: 이름을 대문자로 변환
SELECT
    emp_name,
    upper(emp_name) AS upper_name
FROM employees;

-- 과제 3: 이름을 소문자로 변환
SELECT
    emp_name,
    lower(emp_name) AS lower_name
FROM employees;

-- 과제 4: 부서명 첫 글자만 대문자로
SELECT
    department,
    concat(upper(substr(department, 1, 1)),
lower(substr(department, 2))) AS formatted_dept
FROM employees;
```

练习 1.1: 创建员工信息表

```sql
-- 创建员工表
CREATE TABLE IF NOT EXISTS employees (
    emp_id INT,
    emp_name STRING,
    email STRING,
    phone STRING,
    department STRING,
    salary DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH '/usr/datas/employees.txt'
OVERWRITE INTO TABLE employees;

-- 验证数据
SELECT * FROM employees;
SELECT count(*) as total_count FROM employees;
```

数据文件: employees.txt

练习 1.2: 字符串长度和大小写转换

```sql
-- 任务 1: 计算员工姓名的长度
SELECT
    emp_name,
    length(emp_name) AS name_length
FROM employees;

-- 任务 2: 将姓名转换为大写
SELECT
    emp_name,
    upper(emp_name) AS upper_name
FROM employees;

-- 任务 3: 将姓名转换为小写
SELECT
    emp_name,
    lower(emp_name) AS lower_name
FROM employees;

-- 任务 4: 将部门名称首字母大写
SELECT
    department,
    concat(upper(substr(department, 1, 1)),
lower(substr(department, 2))) AS formatted_dept
FROM employees;
```

```

### 연습 1.3: 문자열 추출과 연결

```sql
-- 과제 5: 이메일에서 사용자명 추출 (@ 앞부분)
SELECT
    email,
    substr(email, 1, instr(email, '@') - 1) AS
username
FROM employees;

-- 과제 6: 직원 ID 와 이름 연결
SELECT
    concat('EMP-', emp_id, ': ', emp_name) AS
emp_info
FROM employees;

-- 과제 7: concat_ws 를 사용하여 여러 필드 연결
SELECT
    concat_ws(' | ', emp_id, emp_name,
department, phone) AS full_info
FROM employees;

-- 과제 8: 직원 이름 역순으로 출력
SELECT
    emp_name,
    reverse(emp_name) AS reversed_name
FROM employees;
```

연습 1.4: 문자열 검색과 치환

```sql
-- 과제 9: 이메일 도메인을 company.com 에서
newcompany.com 으로 변경
SELECT
    email,
    regexp_replace(email, 'company.com',
'newcompany.com') AS new_email
FROM employees;

-- 과제 10: 공백 제거
SELECT
    concat('  ', emp_name, '  ') AS original,
    trim(concat('  ', emp_name, '  ')) AS
trimmed,
    ltrim(concat('  ', emp_name, '  ')) AS
left_trimmed,
    rtrim(concat('  ', emp_name, '  ')) AS
right_trimmed
FROM employees
LIMIT 1;
```

```

### 练习 1.3: 字符串截取和连接

```sql
-- 任务 5: 从邮箱中提取用户名(@ 之前的部分)
SELECT
    email,
    substr(email, 1, instr(email, '@') - 1) AS
username
FROM employees;

-- 任务 6: 连接员工 ID 和姓名
SELECT
    concat('EMP-', emp_id, ': ', emp_name) AS
emp_info
FROM employees;

-- 任务 7: 使用 concat_ws 连接多个字段
SELECT
    concat_ws(' | ', emp_id, emp_name,
department, phone) AS full_info
FROM employees;

-- 任务 8: 反转员工姓名
SELECT
    emp_name,
    reverse(emp_name) AS reversed_name
FROM employees;
```

练习 1.4: 字符串查找和替换

```sql
-- 任务 9: 将邮箱域名从 company.com 改为
newcompany.com
SELECT
    email,
    regexp_replace(email, 'company.com',
'newcompany.com') AS new_email
FROM employees;

-- 任务 10: 去除空格
SELECT
    concat('  ', emp_name, '  ') AS original,
    trim(concat('  ', emp_name, '  ')) AS
trimmed,
    ltrim(concat('  ', emp_name, '  ')) AS
left_trimmed,
    rtrim(concat('  ', emp_name, '  ')) AS
right_trimmed
FROM employees
LIMIT 1;
```

연습 1 정리: 다음 질문에 답하세요

1. `substr`와 `substring` 함수의 차이점은 무엇인가요?
2. 문자열의 마지막 3 글자를 추출하는 방법은?
3. `concat`과 `concat_ws`의 주요 차이점은 무엇인가요?

제 2 부: 날짜 시간 함수 실습 (40 분)

연습 2.1: 주문 테이블 생성

```sql
-- 주문 테이블 생성
CREATE TABLE IF NOT EXISTS orders (
    order_id INT,
    customer_name STRING,
    order_date STRING,
    ship_date STRING,
    order_amount DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH '/usr/datas/orders.txt'
OVERWRITE INTO TABLE orders;

-- 데이터 확인
SELECT * FROM orders;
SELECT count(*) as total_count FROM orders;
```

데이터 파일: orders.txt

연습 2.2: 현재 날짜와 시간 조회

```sql
-- 과제 1: 현재 날짜 조회
SELECT current_date() AS today;

-- 과제 2: 현재 타임스탬프 조회
SELECT current_timestamp() AS now;

-- 과제 3: Unix 타임스탬프 조회
SELECT unix_timestamp() AS unix_time;
```

练习 1 总结: 请回答以下问题

1. `substr` 和 `substring` 函数有什么区别?
2. 如何提取字符串的最后 3 个字符?
3. `concat` 和 `concat_ws` 的主要区别是什么?

第二部分: 日期时间函数实习 (40 分钟)

练习 2.1: 创建订单表

```sql
-- 创建订单表
CREATE TABLE IF NOT EXISTS orders (
    order_id INT,
    customer_name STRING,
    order_date STRING,
    ship_date STRING,
    order_amount DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH '/usr/datas/orders.txt'
OVERWRITE INTO TABLE orders;

-- 验证数据
SELECT * FROM orders;
SELECT count(*) as total_count FROM orders;
```

数据文件: orders.txt

练习 2.2: 获取当前日期和时间

```sql
-- 任务 1: 获取当前日期
SELECT current_date() AS today;

-- 任务 2: 获取当前时间戳
SELECT current_timestamp() AS now;

-- 任务 3: 获取 Unix 时间戳
SELECT unix_timestamp() AS unix_time;
```

```sql
-- 과제 4: Unix 타임스탬프를 날짜로 변환
SELECT from_unixtime(unix_timestamp()) AS
datetime_from_unix;
```

연습 2.3: 날짜 추출과 계산

```sql
-- 과제 5: 주문일자에서 연, 월, 일 추출
SELECT
    order_date,
    year(order_date) AS order_year,
    month(order_date) AS order_month,
    day(order_date) AS order_day,
    weekofyear(order_date) AS week_number
FROM orders;

-- 과제 6: 주문일과 배송일 간 일수 차이 계산
SELECT
    order_id,
    order_date,
    ship_date,
    datediff(ship_date, order_date) AS
days_to_ship
FROM orders;

-- 과제 7: 주문일부터 오늘까지의 일수 계산
SELECT
    order_id,
    order_date,
    datediff(current_date(), order_date) AS
days_since_order
FROM orders;

-- 과제 8: 주문일에 30 일 추가
SELECT
    order_id,
    order_date,
    date_add(order_date, 30) AS due_date
FROM orders;

-- 과제 9: 주문일에서 7 일 빼기
SELECT
    order_id,
    order_date,
    date_sub(order_date, 7) AS preparation_date
FROM orders;
```

연습 2.4: 날짜 형식 변환

```sql
-- 과제 10: 문자열을 날짜로 변환
SELECT
    order_date,
    to_date(order_date) AS date_only
```



```sql
-- 任务 4: 将 Unix 时间戳转换为日期
SELECT from_unixtime(unix_timestamp()) AS
datetime_from_unix;
```

练习 2.3: 日期提取和计算

```sql
-- 任务 5: 从订单日期中提取年、月、日
SELECT
    order_date,
    year(order_date) AS order_year,
    month(order_date) AS order_month,
    day(order_date) AS order_day,
    weekofyear(order_date) AS week_number
FROM orders;

-- 任务 6: 计算订单日期和发货日期之间的天数差
SELECT
    order_id,
    order_date,
    ship_date,
    datediff(ship_date, order_date) AS
days_to_ship
FROM orders;

-- 任务 7: 计算订单日期到今天的天数
SELECT
    order_id,
    order_date,
    datediff(current_date(), order_date) AS
days_since_order
FROM orders;

-- 任务 8: 在订单日期上增加 30 天
SELECT
    order_id,
    order_date,
    date_add(order_date, 30) AS due_date
FROM orders;

-- 任务 9: 从订单日期减去 7 天
SELECT
    order_id,
    order_date,
    date_sub(order_date, 7) AS preperation_date
FROM orders;
```

练习 2.4: 日期格式转换

```sql
-- 任务 10: 将字符串转换为日期
SELECT
    order_date,
    to_date(order_date) AS date_only
```

```
FROM orders;

-- 과제 11: 주문 시간의 각 부분 추출
SELECT
    order_date,
    hour(concat(order_date, ' 14:30:25')) AS
hour_part,
    minute(concat(order_date, ' 14:30:25')) AS
minute_part,
    second(concat(order_date, ' 14:30:25')) AS
second_part
FROM orders
LIMIT 1;

-- 과제 12: 각 주문이 월말까지 남은 일수 계산
SELECT
    order_id,
    order_date,
    last_day(order_date) AS month_end,
    datediff(last_day(order_date), order_date) AS
days_to_month_end
FROM orders;
```

연습 2 정리: 다음 질문에 답하세요

1. `datediff` 함수가 반환하는 단위는 무엇인가요?
2. 특정 날짜가 속한 월의 첫째 날을 구하는 방법은?
3. `current_date()`와 `current_timestamp()`의 차이는 무엇인가요?

제 3 부: 수학 함수 실습 (30 분)

연습 3.1: 제품 판매 테이블 생성

```sql
-- 제품 판매 테이블 생성
CREATE TABLE IF NOT EXISTS product_sales (
    product_id INT,
    product_name STRING,
    unit_price DOUBLE,
    quantity INT,
    discount_rate DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/product_sales.txt'
OVERWRITE INTO TABLE product_sales;
```

```
FROM orders;

-- 任务 11: 提取订单时间的各个部分
SELECT
    order_date,
    hour(concat(order_date, ' 14:30:25')) AS
hour_part,
    minute(concat(order_date, ' 14:30:25')) AS
minute_part,
    second(concat(order_date, ' 14:30:25')) AS
second_part
FROM orders
LIMIT 1;

-- 任务 12: 计算每个订单距离月底还有多少天
SELECT
    order_id,
    order_date,
    last_day(order_date) AS month_end,
    datediff(last_day(order_date), order_date) AS
days_to_month_end
FROM orders;
```

练习 2 总结: 请回答以下问题

1. `datediff` 函数返回的是什么单位?
2. 如何获取某个日期所在月份的第一天?
3. `current_date()` 和 `current_timestamp()` 的区别是什么?

第三部分: 数学函数实习 (30 分钟)

练习 3.1: 创建产品销售表

```sql
-- 创建产品销售表
CREATE TABLE IF NOT EXISTS product_sales (
    product_id INT,
    product_name STRING,
    unit_price DOUBLE,
    quantity INT,
    discount_rate DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/product_sales.txt'
OVERWRITE INTO TABLE product_sales;
```

```sql
-- 데이터 확인
SELECT * FROM product_sales;
SELECT count(*) as total_count FROM
product_sales;
```

데이터 파일: product_sales.txt

연습 3.2: 기본 수학 연산

```sql
-- 과제 1: 판매 총액 계산 (단가 × 수량)
SELECT
    product_name,
    unit_price,
    quantity,
    unit_price * quantity AS total_sales
FROM product_sales;

-- 과제 2: 할인 후 가격 계산
SELECT
    product_name,
    unit_price,
    discount_rate,
    round(unit_price * (1 - discount_rate), 2) AS
discounted_price
FROM product_sales;

-- 과제 3: round 함수로 다양한 소수점 자리수 적용
SELECT
    product_name,
    unit_price,
    round(unit_price) AS rounded_0,
    round(unit_price, 1) AS rounded_1,
    round(unit_price, 2) AS rounded_2
FROM product_sales;
```

연습 3.3: 올림과 내림

```sql
-- 과제 4: floor 로 내림
SELECT
    product_name,
    unit_price,
    floor(unit_price) AS floored_price
FROM product_sales;

-- 과제 5: ceil 로 올림
SELECT
    product_name,
    unit_price,
    ceil(unit_price) AS ceiled_price
FROM product_sales;
```

```sql
-- 验证数据
SELECT * FROM product_sales;
SELECT count(*) as total_count FROM
product_sales;
```

数据文件: product_sales.txt

练习 3.2: 基础数学运算

```sql
-- 任务 1: 计算销售总额(单价 × 数量)
SELECT
    product_name,
    unit_price,
    quantity,
    unit_price * quantity AS total_sales
FROM product_sales;

-- 任务 2: 计算折扣后的价格
SELECT
    product_name,
    unit_price,
    discount_rate,
    round(unit_price * (1 - discount_rate), 2) AS
discounted_price
FROM product_sales;

-- 任务 3: 使用 round 函数保留不同小数位
SELECT
    product_name,
    unit_price,
    round(unit_price) AS rounded_0,
    round(unit_price, 1) AS rounded_1,
    round(unit_price, 2) AS rounded_2
FROM product_sales;
```

练习 3.3: 向上和向下取整

```sql
-- 任务 4: 使用 floor 向下取整
SELECT
    product_name,
    unit_price,
    floor(unit_price) AS floored_price
FROM product_sales;

-- 任务 5: 使用 ceil 向上取整
SELECT
    product_name,
    unit_price,
    ceil(unit_price) AS ceiled_price
FROM product_sales;
```

```sql
-- 과제 6: 평균 단가 계산 및 반올림
SELECT
    round(avg(unit_price), 2) AS avg_price,
    floor(avg(unit_price)) AS avg_floor,
    ceil(avg(unit_price)) AS avg_ceil
FROM product_sales;
```

연습 3.4: 기타 수학 함수

```sql
-- 과제 7: 절댓값 계산
SELECT
    product_name,
    discount_rate,
    abs(discount_rate - 0.10) AS
diff_from_10percent
FROM product_sales;

-- 과제 8: 난수 생성 (추첨용)
SELECT
    product_id,
    product_name,
    rand() AS random_number,
    round(rand() * 100) AS lottery_number
FROM product_sales;

-- 과제 9: 판매액의 합계, 최댓값, 최솟값 계산
SELECT
    sum(unit_price * quantity) AS total_revenue,
    max(unit_price * quantity) AS max_revenue,
    min(unit_price * quantity) AS min_revenue,
    avg(unit_price * quantity) AS avg_revenue
FROM product_sales;
```

연습 3 정리: 다음 질문에 답하세요

1. `round`, `floor`, `ceil`의 차이점은 무엇인가요?
2. 1 부터 100 사이의 난수 정수를 생성하는 방법은?
3. `abs` 함수의 일반적인 활용 사례는 무엇인가요?

제 4 부: 조건 함수 실습 (40 분)

연습 4.1: 학생 성적 테이블 생성

```sql
-- 학생 성적 테이블 생성
CREATE TABLE IF NOT EXISTS student_scores (
    student_id INT,
    student_name STRING,
```

```sql
-- 任务 6: 计算平均单价并取整
SELECT
    round(avg(unit_price), 2) AS avg_price,
    floor(avg(unit_price)) AS avg_floor,
    ceil(avg(unit_price)) AS avg_ceil
FROM product_sales;
```

练习 3.4: 其他数学函数

```sql
-- 任务 7: 计算绝对值
SELECT
    product_name,
    discount_rate,
    abs(discount_rate - 0.10) AS
diff_from_10percent
FROM product_sales;

-- 任务 8: 生成随机数(用于抽奖)
SELECT
    product_id,
    product_name,
    rand() AS random_number,
    round(rand() * 100) AS lottery_number
FROM product_sales;

-- 任务 9: 计算销售额的总和、最大值、最小值
SELECT
    sum(unit_price * quantity) AS total_revenue,
    max(unit_price * quantity) AS max_revenue,
    min(unit_price * quantity) AS min_revenue,
    avg(unit_price * quantity) AS avg_revenue
FROM product_sales;
```

练习 3 总结: 请回答以下问题

1. `round`, `floor`, 和 `ceil` 的区别是什么?
2. 如何生成 1 到 100 之间的随机整数?
3. `abs` 函数的常见应用场景有哪些?

第四部分: 条件函数实习 (40 分钟)

练习 4.1: 创建学生成绩表

```sql
-- 创建学生成绩表
CREATE TABLE IF NOT EXISTS student_scores (
    student_id INT,
    student_name STRING,
```

```
    math_score INT,
    english_score INT,
    computer_score INT,
    gender STRING
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/student_scores.txt'
OVERWRITE INTO TABLE student_scores;

-- 데이터 확인
SELECT * FROM student_scores;
SELECT count(*) as total_count FROM
student_scores;
```

데이터 파일: student_scores.txt

연습 4.2: IF 조건 판단

```sql
-- 과제 1: 수학 점수 합격 여부 판단 (60 점 이상 합격)
SELECT
    student_name,
    math_score,
    if(math_score >= 60, '합격', '불합격') AS
math_status
FROM student_scores;

-- 과제 2: 성별에 따라 다른 호칭 표시
SELECT
    student_name,
    gender,
    if(gender = 'M', '군', '양') AS title
FROM student_scores;

-- 과제 3: 총점이 240 점 이상인지 판단 (우수 기준선)
SELECT
    student_name,
    math_score + english_score + computer_score
AS total_score,
    if(math_score + english_score +
computer_score >= 240, '우수', '양호') AS
performance
FROM student_scores
WHERE math_score IS NOT NULL;
```

연습 4.3: CASE WHEN 다중 조건 판단

```sql
```

```
    math_score INT,
    english_score INT,
    computer_score INT,
    gender STRING
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/student_scores.txt'
OVERWRITE INTO TABLE student_scores;

-- 验证数据
SELECT * FROM student_scores;
SELECT count(*) as total_count FROM
student_scores;
```

数据文件: student_scores.txt

练习 4.2: IF 条件判断

```sql
-- 任务 1: 判断数学成绩是否及格(60 分及格)
SELECT
    student_name,
    math_score,
    if(math_score >= 60, '及格', '不及格') AS
math_status
FROM student_scores;

-- 任务 2: 根据性别显示不同的称呼
SELECT
    student_name,
    gender,
    if(gender = 'M', '先生', '女士') AS title
FROM student_scores;

-- 任务 3: 判断总分是否超过 240 分(优秀线)
SELECT
    student_name,
    math_score + english_score + computer_score
AS total_score,
    if(math_score + english_score +
computer_score >= 240, '优秀', '良好') AS
performance
FROM student_scores
WHERE math_score IS NOT NULL;
```

练习 4.3: CASE WHEN 多条件判断

```sql
```

```sql
-- 과제 4: 수학 성적에 따른 등급 구분
SELECT
    student_name,
    math_score,
    CASE
        WHEN math_score >= 90 THEN 'A 우수'
        WHEN math_score >= 80 THEN 'B 양호'
        WHEN math_score >= 70 THEN 'C 보통'
        WHEN math_score >= 60 THEN 'D 합격'
        ELSE 'F 불합격'
    END AS grade
FROM student_scores;

-- 과제 5: 평균 점수에 따른 학생 분류
SELECT
    student_name,
    round((math_score + english_score +
computer_score) / 3.0, 2) AS avg_score,
    CASE
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 90 THEN '최우수'
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 80 THEN '우등생'
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 70 THEN '보통'
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 60 THEN '합격'
        ELSE '노력 필요'
    END AS student_type
FROM student_scores
WHERE math_score IS NOT NULL;
```

연습 4.4: NULL 값 처리 함수

```sql
-- 과제 6: ISNULL 로 NULL 값 판단
SELECT
    student_name,
    math_score,
    isnull(math_score) AS is_null
FROM student_scores;

-- 과제 7: COALESCE 로 NULL 처리 (0 으로 대체)
SELECT
    student_name,
    math_score,
    coalesce(math_score, 0) AS math_score_fixed
FROM student_scores;

-- 과제 8: NVL 로 NULL 처리 후 평균 계산
SELECT
    student_name,
    nvl(math_score, 0) AS math,
```

```sql
-- 任务 4: 根据数学成绩划分等级
SELECT
    student_name,
    math_score,
    CASE
        WHEN math_score >= 90 THEN 'A 优秀'
        WHEN math_score >= 80 THEN 'B 良好'
        WHEN math_score >= 70 THEN 'C 中等'
        WHEN math_score >= 60 THEN 'D 及格'
        ELSE 'E 不及格'
    END AS grade
FROM student_scores;

-- 任务 5: 根据平均分划分学生类别
SELECT
    student_name,
    round((math_score + english_score +
computer_score) / 3.0, 2) AS avg_score,
    CASE
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 90 THEN '学霸'
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 80 THEN '优等生'
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 70 THEN '中等生'
        WHEN (math_score + english_score +
computer_score) / 3.0 >= 60 THEN '及格'
        ELSE '需要努力'
    END AS student_type
FROM student_scores
WHERE math_score IS NOT NULL;
```

练习 4.4: NULL 值处理函数

```sql
-- 任务 6: 使用 ISNULL 判断空值
SELECT
    student_name,
    math_score,
    isnull(math_score) AS is_null
FROM student_scores;

-- 任务 7: 使用 COALESCE 处理空值(用 0 替代)
SELECT
    student_name,
    math_score,
    coalesce(math_score, 0) AS math_score_fixed
FROM student_scores;

-- 任务 8: 使用 NVL 处理空值并计算平均分
SELECT
    student_name,
    nvl(math_score, 0) AS math,
```

```
    english_score,
    computer_score,
    round((nvl(math_score, 0) + english_score +
computer_score) / 3.0, 2) AS avg_score
FROM student_scores;

-- 과제 9: NULLIF 로 두 값 비교
SELECT
    student_name,
    math_score,
    english_score,
    nullif(math_score, english_score) AS
score_diff
FROM student_scores
WHERE math_score IS NOT NULL;
```

연습 4 정리: 다음 질문에 답하세요

1. `IF`와 `CASE WHEN`의 사용 시나리오 차이는 무엇인가요?
2. `COALESCE`와 `NVL`의 차이점은 무엇인가요?
3. 필드가 NULL 인지 확인하는 방법은?

제 5 부: 집합 함수 실습 (30 분)

연습 5.1: 사용자 관심사 테이블 생성

```sql
-- 사용자 관심사 테이블 생성 (복합 타입 포함)
CREATE TABLE IF NOT EXISTS user_interests (
    user_id INT,
    user_name STRING,
    hobbies ARRAY<STRING>,
    scores MAP<STRING, INT>
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/user_interests.txt'
OVERWRITE INTO TABLE user_interests;

-- 데이터 확인
SELECT * FROM user_interests;
SELECT count(*) as total_count FROM
user_interests;
```

```
    english_score,
    computer_score,
    round((nvl(math_score, 0) + english_score +
computer_score) / 3.0, 2) AS avg_score
FROM student_scores;

-- 任务 9: 使用 NULLIF 比较两个值
SELECT
    student_name,
    math_score,
    english_score,
    nullif(math_score, english_score) AS
score_diff
FROM student_scores
WHERE math_score IS NOT NULL;
```

练习 4 总结: 请回答以下问题

1. `IF` 和 `CASE WHEN` 的使用场景有什么区别?
2. `COALESCE` 和 `NVL` 有什么区别?
3. 如何检查一个字段是否为 NULL?

第五部分: 集合函数实习 (30 分钟)

练习 5.1: 创建用户兴趣表

```sql
-- 创建用户兴趣表(包含复杂类型)
CREATE TABLE IF NOT EXISTS user_interests (
    user_id INT,
    user_name STRING,
    hobbies ARRAY<STRING>,
    scores MAP<STRING, INT>
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/user_interests.txt'
OVERWRITE INTO TABLE user_interests;

-- 验证数据
SELECT * FROM user_interests;
SELECT count(*) as total_count FROM
user_interests;
```

데이터 파일: user_interests.txt

연습 5.2: 배열 함수

```sql
-- 과제 1: 취미 개수 조회
SELECT
    user_name,
    hobbies,
    size(hobbies) AS hobby_count
FROM user_interests;

-- 과제 2: 특정 취미 포함 여부 확인
SELECT
    user_name,
    hobbies,
    array_contains(hobbies, '독서') AS
likes_reading
FROM user_interests;

-- 과제 3: 취미 배열 정렬
SELECT
    user_name,
    hobbies AS original_hobbies,
    sort_array(hobbies) AS sorted_hobbies
FROM user_interests;
```

연습 5.3: Map 함수

```sql
-- 과제 4: Map 크기 조회
SELECT
    user_name,
    scores,
    size(scores) AS score_count
FROM user_interests;

-- 과제 5: Map 의 모든 키 조회
SELECT
    user_name,
    scores,
    map_keys(scores) AS subjects
FROM user_interests;

-- 과제 6: Map 의 모든 값 조회
SELECT
    user_name,
    scores,
    map_values(scores) AS score_values
FROM user_interests;

-- 과제 7: Map 에서 특정 키의 값 조회
SELECT
```

数据文件: user_interests.txt

练习 5.2: 数组函数

```sql
-- 任务 1: 获取爱好数量
SELECT
    user_name,
    hobbies,
    size(hobbies) AS hobby_count
FROM user_interests;

-- 任务 2: 检查是否包含特定爱好
SELECT
    user_name,
    hobbies,
    array_contains(hobbies, 'reading') AS
likes_reading
FROM user_interests;

-- 任务 3: 对爱好数组进行排序
SELECT
    user_name,
    hobbies AS original_hobbies,
    sort_array(hobbies) AS sorted_hobbies
FROM user_interests;
```

练习 5.3: Map 函数

```sql
-- 任务 4: 获取 Map 的大小
SELECT
    user_name,
    scores,
    size(scores) AS score_count
FROM user_interests;

-- 任务 5: 获取 Map 的所有键
SELECT
    user_name,
    scores,
    map_keys(scores) AS subjects
FROM user_interests;

-- 任务 6: 获取 Map 的所有值
SELECT
    user_name,
    scores,
    map_values(scores) AS score_values
FROM user_interests;

-- 任务 7: 从 Map 中获取特定键的值
SELECT
```

```
    user_name,
    scores,
    scores['수학'] AS math_score,
    scores['영어'] AS english_score
FROM user_interests;
```

연습 5 정리: 다음 질문에 답하세요

1. `size()` 함수는 어떤 데이터 타입에 사용할 수 있나요?
2. 배열에 특정 요소가 포함되어 있는지 확인하는 방법은?
3. Map 과 Array 의 주요 차이점은 무엇인가요?

제 6 부: 집계 함수 실습 (40 분)

연습 6.1: 판매 상세 테이블 생성

```sql
-- 판매 상세 테이블 생성
CREATE TABLE IF NOT EXISTS sales_detail (
    sale_id INT,
    sale_date STRING,
    region STRING,
    salesperson STRING,
    product_category STRING,
    amount DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/sales_detail.txt'
OVERWRITE INTO TABLE sales_detail;

-- 데이터 확인
SELECT * FROM sales_detail;
SELECT count(*) as total_count FROM sales_detail;
```

데이터 파일: sales_detail.txt

연습 6.2: 기본 집계 함수

```sql
-- 과제 1: 총 판매액 계산
```



```
    user_name,
    scores,
    scores['math'] AS math_score,
    scores['english'] AS english_score
FROM user_interests;
```

练习 5 总结: 请回答以下问题

1. `size()` 函数可以用于哪些数据类型?
2. 如何检查数组中是否包含某个元素?
3. Map 和 Array 的主要区别是什么?

第六部分: 聚合函数实习 (40 分钟)

练习 6.1: 创建销售明细表

```sql
-- 创建销售明细表
CREATE TABLE IF NOT EXISTS sales_detail (
    sale_id INT,
    sale_date STRING,
    region STRING,
    salesperson STRING,
    product_category STRING,
    amount DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/sales_detail.txt'
OVERWRITE INTO TABLE sales_detail;

-- 验证数据
SELECT * FROM sales_detail;
SELECT count(*) as total_count FROM sales_detail;
```

数据文件: sales_detail.txt

练习 6.2: 基础聚合函数

```sql
-- 任务 1: 计算总销售额
```

```sql
SELECT
    sum(amount) AS total_sales
FROM sales_detail;

-- 과제 2: 평균 판매액 계산
SELECT
    round(avg(amount), 2) AS avg_sales
FROM sales_detail;

-- 과제 3: 최대 및 최소 판매액 조회
SELECT
    max(amount) AS max_sales,
    min(amount) AS min_sales
FROM sales_detail;

-- 과제 4: 판매 기록 수 집계
SELECT
    count(*) AS total_records,
    count(DISTINCT region) AS unique_regions,
    count(DISTINCT salesperson) AS
unique_salespeople
FROM sales_detail;
```

연습 6.3: 그룹별 집계

```sql
-- 과제 5: 지역별 판매액 통계
SELECT
    region,
    count(*) AS sale_count,
    sum(amount) AS total_sales,
    round(avg(amount), 2) AS avg_sales
FROM sales_detail
GROUP BY region
ORDER BY total_sales DESC;

-- 과제 6: 판매원별 실적 통계
SELECT
    salesperson,
    count(*) AS sale_count,
    sum(amount) AS total_sales,
    max(amount) AS max_single_sale,
    min(amount) AS min_single_sale
FROM sales_detail
GROUP BY salesperson
ORDER BY total_sales DESC;

-- 과제 7: 제품 카테고리별 통계
SELECT
    product_category,
    count(*) AS sale_count,
    sum(amount) AS total_sales,
    round(avg(amount), 2) AS avg_sales
FROM sales_detail
GROUP BY product_category;
```

```sql
SELECT
    sum(amount) AS total_sales
FROM sales_detail;

-- 任务 2: 计算平均销售额
SELECT
    round(avg(amount), 2) AS avg_sales
FROM sales_detail;

-- 任务 3: 找出最大和最小销售额
SELECT
    max(amount) AS max_sales,
    min(amount) AS min_sales
FROM sales_detail;

-- 任务 4: 统计销售记录数
SELECT
    count(*) AS total_records,
    count(DISTINCT region) AS unique_regions,
    count(DISTINCT salesperson) AS
unique_salespeople
FROM sales_detail;
```

练习 6.3: 分组聚合

```sql
-- 任务 5: 按地区统计销售额
SELECT
    region,
    count(*) AS sale_count,
    sum(amount) AS total_sales,
    round(avg(amount), 2) AS avg_sales
FROM sales_detail
GROUP BY region
ORDER BY total_sales DESC;

-- 任务 6: 按销售人员统计业绩
SELECT
    salesperson,
    count(*) AS sale_count,
    sum(amount) AS total_sales,
    max(amount) AS max_single_sale,
    min(amount) AS min_single_sale
FROM sales_detail
GROUP BY salesperson
ORDER BY total_sales DESC;

-- 任务 7: 按产品类别统计
SELECT
    product_category,
    count(*) AS sale_count,
    sum(amount) AS total_sales,
    round(avg(amount), 2) AS avg_sales
FROM sales_detail
GROUP BY product_category;
```

연습 6.4: COLLECT 함수

```sql
-- 과제 8: 각 지역의 모든 판매액 수집 (중복 제거)
SELECT
    region,
    collect_set(cast(amount AS INT)) AS
unique_amounts
FROM sales_detail
GROUP BY region;

-- 과제 9: 각 판매원의 모든 판매 날짜 수집 (중복
유지)
SELECT
    salesperson,
    collect_list(sale_date) AS all_sale_dates
FROM sales_detail
GROUP BY salesperson;

-- 과제 10: 종합 통계 - 지역별 판매원 수집
SELECT
    region,
    collect_set(salesperson) AS salespeople,
    count(*) AS sale_count,
    sum(amount) AS total_sales
FROM sales_detail
GROUP BY region;
```

연습 6 정리: 다음 질문에 답하세요

1. `collect_set`과 `collect_list`의 차이점은 무엇인가요?
2. 특정 필드의 고유 값 개수를 계산하는 방법은?
3. GROUP BY 절의 역할은 무엇인가요?

제 7 부: 데이터 마스킹 함수 실습 (30 분)

연습 7.1: 고객 정보 테이블 생성

```sql
-- 고객 정보 테이블 생성
CREATE TABLE IF NOT EXISTS customer_info (
    customer_id INT,
    customer_name STRING,
    id_card STRING,
    phone STRING,
    email STRING,
    credit_card STRING
) ROW FORMAT DELIMITED
```

练习 6.4: COLLECT 函数

```sql
-- 任务 8: 收集每个地区的所有销售额(去重)
SELECT
    region,
    collect_set(cast(amount AS INT)) AS
unique_amounts
FROM sales_detail
GROUP BY region;

-- 任务 9: 收集每个销售人员的所有销售日期(保留重复)

SELECT
    salesperson,
    collect_list(sale_date) AS all_sale_dates
FROM sales_detail
GROUP BY salesperson;

-- 任务 10: 综合统计-按地区收集销售人员
SELECT
    region,
    collect_set(salesperson) AS salespeople,
    count(*) AS sale_count,
    sum(amount) AS total_sales
FROM sales_detail
GROUP BY region;
```

练习 6 总结: 请回答以下问题

1. `collect_set` 和 `collect_list` 的区别是什么?
2. 如何计算某个字段的不重复值数量?
3. GROUP BY 子句的作用是什么?

第七部分: 数据脱敏函数实习 (30 分钟)

练习 7.1: 创建客户信息表

```sql
-- 创建客户信息表
CREATE TABLE IF NOT EXISTS customer_info (
    customer_id INT,
    customer_name STRING,
    id_card STRING,
    phone STRING,
    email STRING,
    credit_card STRING
) ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/customer_info.txt'
OVERWRITE INTO TABLE customer_info;

-- 데이터 확인
SELECT * FROM customer_info;
SELECT count(*) as total_count FROM
customer_info;
```

데이터 파일: customer_info.txt

연습 7.2: 전화번호 마스킹

```sql
-- 과제 1: mask_show_first_n 으로 앞 3 자리 표시
SELECT
    customer_name,
    phone,
    mask_show_first_n(phone, 3) AS
masked_phone_first3
FROM customer_info;

-- 과제 2: mask_show_last_n 으로 뒤 4 자리 표시
SELECT
    customer_name,
    phone,
    mask_show_last_n(phone, 4) AS
masked_phone_last4
FROM customer_info;

-- 과제 3: mask_first_n 으로 앞 3 자리 숨김
SELECT
    customer_name,
    phone,
    mask_first_n(phone, 3) AS
masked_phone_hide_first3
FROM customer_info;

-- 과제 4: mask_last_n 으로 뒤 4 자리 숨김
SELECT
    customer_name,
    phone,
    mask_last_n(phone, 4) AS
masked_phone_hide_last4
FROM customer_info;
```

연습 7.3: 주민등록번호 마스킹

```sql
```

```
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/customer_info.txt'
OVERWRITE INTO TABLE customer_info;

-- 验证数据
SELECT * FROM customer_info;
SELECT count(*) as total_count FROM
customer_info;
```

数据文件: customer_info.txt

练习 7.2: 手机号脱敏

```sql
-- 任务 1: 使用 mask_show_first_n 显示前 3 位
SELECT
    customer_name,
    phone,
    mask_show_first_n(phone, 3) AS
masked_phone_first3
FROM customer_info;

-- 任务 2: 使用 mask_show_last_n 显示后 4 位
SELECT
    customer_name,
    phone,
    mask_show_last_n(phone, 4) AS
masked_phone_last4
FROM customer_info;

-- 任务 3: 使用 mask_first_n 掩盖前 3 位
SELECT
    customer_name,
    phone,
    mask_first_n(phone, 3) AS
masked_phone_hide_first3
FROM customer_info;

-- 任务 4: 使用 mask_last_n 掩盖后 4 位
SELECT
    customer_name,
    phone,
    mask_last_n(phone, 4) AS
masked_phone_hide_last4
FROM customer_info;
```

练习 7.3: 身份证号脱敏

```sql
```

```sql
-- 과제 5: 주민번호 앞 6 자리와 뒤 1 자리만 표시
SELECT
    customer_name,
    id_card,
    concat(
        substr(id_card, 1, 6),
        '-*******'
        substr(id_card, -4)
    ) AS masked_id_card
FROM customer_info;

-- 과제 6: mask 함수로 완전 마스킹
SELECT
    customer_name,
    id_card,
    mask(id_card) AS fully_masked_id
FROM customer_info;
```

연습 7.4: 이메일과 카드번호 마스킹

```sql
-- 과제 7: 이메일 마스킹 (@ 뒤 도메인 유지)
SELECT
    customer_name,
    email,
    regexp_replace(email,
'^([^@]{2})[^@]+(@.+)$', '$1****$2') AS
masked_email
FROM customer_info;

-- 과제 8: 카드번호 마스킹 (앞 4 자리와 뒤 4 자리
표시)
SELECT
    customer_name,
    credit_card,
    concat(
        substr(credit_card, 1, 4),
        '-****-****-',
        substr(credit_card, -4)
    ) AS masked_card
FROM customer_info;

-- 과제 9: 종합 마스킹 쿼리
SELECT
    customer_id,
    customer_name,
    concat(substr(id_card, 1, 6), '-*******') AS
masked_id_card,
    concat(substr(phone, 1, 3), '****',
substr(phone, -4)) AS masked_phone,
    regexp_replace(email,
'^([^@]{2})[^@]+(@.+)$', '$1****$2') AS
masked_email,
    concat(substr(credit_card, 1, 4), '-****-
****-', substr(credit_card, -4)) AS masked_card
FROM customer_info;
```

```sql
-- 任务 5: 只显示身份证前 6 位和后 4 位
SELECT
    customer_name,
    id_card,
    concat(
        substr(id_card, 1, 6),
        '********',
        substr(id_card, -4)
    ) AS masked_id_card
FROM customer_info;

-- 任务 6: 使用 mask 函数完全掩码
SELECT
    customer_name,
    id_card,
    mask(id_card) AS fully_masked_id
FROM customer_info;
```

练习 7.4: 邮箱和银行卡脱敏

```sql
-- 任务 7: 邮箱脱敏(保留@后的域名)
SELECT
    customer_name,
    email,
    regexp_replace(email,
'^([^@]{2})[^@]+(@.+)$', '$1****$2') AS
masked_email
FROM customer_info;

-- 任务 8: 银行卡号脱敏(显示前 6 位和后 4 位)

SELECT
    customer_name,
    credit_card,
    concat(
        substr(credit_card, 1, 6),
        '****',
        substr(credit_card, -4)
    ) AS masked_card
FROM customer_info;

-- 任务 9: 综合脱敏查询
SELECT
    customer_id,
    customer_name,
    concat(substr(id_card, 1, 6), '********',
substr(id_card, -4)) AS masked_id_card,
    concat(substr(phone, 1, 3), '****',
substr(phone, -4)) AS masked_phone,
    regexp_replace(email,
'^([^@]{2})[^@]+(@.+)$', '$1****$2') AS
masked_email,
    concat(substr(credit_card, 1, 6), '****',
substr(credit_card, -4)) AS masked_card
FROM customer_info;
```

연습 7 정리: 다음 질문에 답하세요

1. 민감한 데이터를 마스킹 처리해야 하는 이유는?
2. `mask_show_first_n`과 `mask_first_n`의 차이점은 무엇인가요?
3. 사용자 정의 마스킹 규칙을 만드는 방법은?

제 8 부: 타입 변환 함수 실습 (20 분)

연습 8.1: 혼합 타입 데이터 테이블 생성

```sql
-- 테스트 데이터 테이블 생성
CREATE TABLE IF NOT EXISTS type_conversion_test (
    id INT,
    str_number STRING,
    str_decimal STRING,
    int_value INT,
    double_value DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/type_conversion_test.txt'
OVERWRITE INTO TABLE type_conversion_test;

-- 데이터 확인
SELECT * FROM type_conversion_test;
SELECT count(*) as total_count FROM
type_conversion_test;
```

데이터 파일: type_conversion_test.txt

연습 8.2: CAST 함수 활용

```sql
-- 과제 1: 문자열을 정수로 변환
SELECT
    str_number,
    cast(str_number AS INT) AS int_number,
    typeof(cast(str_number AS INT)) AS data_type
FROM type_conversion_test;

-- 과제 2: 문자열을 소수로 변환
SELECT
```

练习 7 总结: 请回答以下问题

1. 为什么需要对敏感数据进行脱敏处理?
2. `mask_show_first_n` 和 `mask_first_n` 的区别是什么?
3. 如何自定义脱敏规则?

第八部分: 类型转换函数实习 (20 分钟)

练习 8.1: 创建混合类型数据表

```sql
-- 创建测试数据表
CREATE TABLE IF NOT EXISTS type_conversion_test (
    id INT,
    str_number STRING,
    str_decimal STRING,
    int_value INT,
    double_value DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/type_conversion_test.txt'
OVERWRITE INTO TABLE type_conversion_test;

-- 验证数据
SELECT * FROM type_conversion_test;
SELECT count(*) as total_count FROM
type_conversion_test;
```

数据文件: type_conversion_test.txt

练习 8.2: CAST 函数应用

```sql
-- 任务 1: 将字符串转换为整数
SELECT
    str_number,
    cast(str_number AS INT) AS int_number,
    typeof(cast(str_number AS INT)) AS data_type
FROM type_conversion_test;

-- 任务 2: 将字符串转换为小数
SELECT
```

```
    str_decimal,
    cast(str_decimal AS DOUBLE) AS double_number,
    typeof(cast(str_decimal AS DOUBLE)) AS
data_type
FROM type_conversion_test;

-- 과제 3: 정수를 문자열로 변환
SELECT
    int_value,
    cast(int_value AS STRING) AS str_value,
    concat('값: ', cast(int_value AS STRING)) AS
formatted_value
FROM type_conversion_test;

-- 과제 4: 소수를 정수로 변환 (반올림)
SELECT
    double_value,
    cast(double_value AS INT) AS int_value,
    round(double_value) AS rounded_value
FROM type_conversion_test;

-- 과제 5: 종합 타입 변환
SELECT
    id,
    cast(str_number AS BIGINT) + int_value AS
sum_result,
    cast(str_decimal AS DOUBLE) * double_value AS
product_result
FROM type_conversion_test;
```

연습 8 정리: 다음 질문에 답하세요

1. CAST 함수의 주요 용도는 무엇인가요?
2. DOUBLE 을 INT 로 변환할 때 어떤 일이 발생하나요?
3. 변환 후 데이터 타입을 확인하는 방법은?

제 9 부: 종합 실전 사례 (60 분)

사례 1: 전자상거래 주문 분석 시스템

```sql
-- 전자상거래 주문 테이블 생성
CREATE TABLE IF NOT EXISTS ecommerce_orders (
    order_id STRING,
    customer_name STRING,
    phone STRING,
    order_date STRING,
    product_name STRING,
    quantity INT,
    unit_price DOUBLE,
```

```
    str_decimal,
    cast(str_decimal AS DOUBLE) AS double_number,
    typeof(cast(str_decimal AS DOUBLE)) AS
data_type
FROM type_conversion_test;

-- 任务 3: 将整数转换为字符串
SELECT
    int_value,
    cast(int_value AS STRING) AS str_value,
    concat('Value: ', cast(int_value AS STRING))
AS formatted_value
FROM type_conversion_test;

-- 任务 4: 将小数转换为整数(四舍五入)
SELECT
    double_value,
    cast(double_value AS INT) AS int_value,
    round(double_value) AS rounded_value
FROM type_conversion_test;

-- 任务 5: 综合类型转换
SELECT
    id,
    cast(str_number AS BIGINT) + int_value AS
sum_result,
    cast(str_decimal AS DOUBLE) * double_value AS
product_result
FROM type_conversion_test;
```

练习 8 总结: 请回答以下问题

1. CAST 函数的主要用途是什么?
2. 将 DOUBLE 转换为 INT 时会发生什么?
3. 如何验证转换后的数据类型?

第九部分: 综合实战案例 (60 分钟)

案例 1: 电商订单分析系统

```sql
-- 创建电商订单表
CREATE TABLE IF NOT EXISTS ecommerce_orders (
    order_id STRING,
    customer_name STRING,
    phone STRING,
    order_date STRING,
    product_name STRING,
    quantity INT,
    unit_price DOUBLE,
```

```
    region STRING,
    status STRING
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/ecommerce_orders.txt'
OVERWRITE INTO TABLE ecommerce_orders;

-- 데이터 확인
SELECT * FROM ecommerce_orders;
SELECT count(*) as total_count FROM
ecommerce_orders;
```

데이터 파일: ecommerce_orders.txt

과제 1: 주문 기본 정보 처리

```sql
-- 1.1 주문 날짜 추출 (년-월-일)
SELECT
    order_id,
    order_date,
    to_date(order_date) AS order_date_only,
    year(order_date) AS order_year,
    month(order_date) AS order_month,
    day(order_date) AS order_day
FROM ecommerce_orders;

-- 1.2 주문 총액 계산 및 형식화
SELECT
    order_id,
    product_name,
    quantity,
    unit_price,
    quantity * unit_price AS total_amount,
    concat('₩', format_number(quantity *
unit_price, 0)) AS formatted_amount
FROM ecommerce_orders;

-- 1.3 고객 전화번호 마스킹
SELECT
    order_id,
    customer_name,
    phone,
    concat(substr(phone, 1, 3), '****',
substr(phone, -4)) AS masked_phone
FROM ecommerce_orders;
```

과제 2: 주문 통계 분석

```sql
```


```
    region STRING,
    status STRING
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/ecommerce_orders.txt'
OVERWRITE INTO TABLE ecommerce_orders;

-- 验证数据
SELECT * FROM ecommerce_orders;
SELECT count(*) as total_count FROM
ecommerce_orders;
```

数据文件: ecommerce_orders.txt

任务 1: 订单基本信息处理

```sql
-- 1.1 提取订单日期(年-月-日)
SELECT
    order_id,
    order_date,
    to_date(order_date) AS order_date_only,
    year(order_date) AS order_year,
    month(order_date) AS order_month,
    day(order_date) AS order_day
FROM ecommerce_orders;

-- 1.2 计算订单总金额并格式化
SELECT
    order_id,
    product_name,
    quantity,
    unit_price,
    quantity * unit_price AS total_amount,
    concat('¥', round(quantity * unit_price, 2))
AS formatted_amount
FROM ecommerce_orders;

-- 1.3 客户手机号脱敏
SELECT
    order_id,
    customer_name,
    phone,
    concat(substr(phone, 1, 3), '****',
substr(phone, -4)) AS masked_phone
FROM ecommerce_orders;
```

任务 2: 订单统计分析

```sql
```

```sql
-- 2.1 지역별 주문 통계
SELECT
    region,
    count(*) AS order_count,
    sum(quantity * unit_price) AS total_revenue,
    round(avg(quantity * unit_price), 2) AS
avg_order_value,
    collect_set(product_name) AS products_sold
FROM ecommerce_orders
WHERE status != '취소됨'
GROUP BY region
ORDER BY total_revenue DESC;

-- 2.2 제품별 판매 현황 통계
SELECT
    product_name,
    count(*) AS order_count,
    sum(quantity) AS total_quantity,
    sum(quantity * unit_price) AS total_revenue,
    round(avg(unit_price), 2) AS avg_unit_price
FROM ecommerce_orders
WHERE status = '배송완료'
GROUP BY product_name
ORDER BY total_revenue DESC;

-- 2.3 상태별 주문 통계
SELECT
    status,
    count(*) AS order_count,
    sum(quantity * unit_price) AS total_amount,
    CASE
        WHEN status = '배송완료' THEN '정상'
        WHEN status = '배송중' THEN '처리중'
        WHEN status = '취소됨' THEN '이상'
        ELSE '미분류'
    END AS status_category
FROM ecommerce_orders
GROUP BY status;
```

과제 3: 고급 분석

```sql
-- 3.1 고액 주문 찾기 (주문금액 > 1,500,000 원)
SELECT
    order_id,
    customer_name,
    product_name,
    quantity * unit_price AS order_amount,
    CASE
        WHEN quantity * unit_price > 3000000 THEN
'VIP 주문'
        WHEN quantity * unit_price > 1500000 THEN
'고액주문'
        ELSE '일반주문'
```

```sql
-- 2.1 按地区统计订单
SELECT
    region,
    count(*) AS order_count,
    sum(quantity * unit_price) AS total_revenue,
    round(avg(quantity * unit_price), 2) AS
avg_order_value,
    collect_set(product_name) AS products_sold
FROM ecommerce_orders
WHERE status != '已取消'
GROUP BY region
ORDER BY total_revenue DESC;

-- 2.2 按产品统计销售情况
SELECT
    product_name,
    count(*) AS order_count,
    sum(quantity) AS total_quantity,
    sum(quantity * unit_price) AS total_revenue,
    round(avg(unit_price), 2) AS avg_unit_price
FROM ecommerce_orders
WHERE status = '已完成'
GROUP BY product_name
ORDER BY total_revenue DESC;

-- 2.3 按状态统计订单
SELECT
    status,
    count(*) AS order_count,
    sum(quantity * unit_price) AS total_amount,
    CASE
        WHEN status = '已完成' THEN '正常'
        WHEN status = '处理中' THEN '待处理'
        WHEN status = '已取消' THEN '异常'
        ELSE '未知'
    END AS status_category
FROM ecommerce_orders
GROUP BY status;
```

任务 3: 高级分析

```sql
-- 3.1 找出高价值订单(订单金额>5000)
SELECT
    order_id,
    customer_name,
    product_name,
    quantity * unit_price AS order_amount,
    CASE
        WHEN quantity * unit_price > 10000 THEN
'VIP 订单'
        WHEN quantity * unit_price > 5000 THEN '
高价值订单'
        ELSE '普通订单'
```

```
    END AS order_level
FROM ecommerce_orders
WHERE quantity * unit_price > 1500000
ORDER BY order_amount DESC;

-- 3.2 주문 경과 일수 계산
SELECT
    order_id,
    customer_name,
    to_date(order_date) AS order_date,
    datediff(current_date(), to_date(order_date))
AS days_since_order,
    CASE
        WHEN datediff(current_date(),
to_date(order_date)) <= 7 THEN '최근 1 주일'
        WHEN datediff(current_date(),
to_date(order_date)) <= 30 THEN '최근 1 개월'
        ELSE '1 개월 이전'
    END AS order_period
FROM ecommerce_orders;

-- 3.3 종합 리포트
SELECT
    region,
    status,
    count(*) AS order_count,
    sum(quantity) AS total_quantity,
    round(sum(quantity * unit_price), 2) AS
total_revenue,
    round(avg(quantity * unit_price), 2) AS
avg_order_value,
    max(quantity * unit_price) AS
max_order_value,
    min(quantity * unit_price) AS min_order_value
FROM ecommerce_orders
GROUP BY region, status
ORDER BY region, total_revenue DESC;
```

사례 2: 직원 급여 분석 시스템

```sql
-- 직원 급여 테이블 생성
CREATE TABLE IF NOT EXISTS employee_salary (
    emp_id STRING,
    emp_name STRING,
    department STRING,
    position STRING,
    hire_date STRING,
    base_salary DOUBLE,
    bonus DOUBLE,
    deduction DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
```

```
    END AS order_level
FROM ecommerce_orders
WHERE quantity * unit_price > 5000
ORDER BY order_amount DESC;

-- 3.2 计算订单距今天数
SELECT
    order_id,
    customer_name,
    to_date(order_date) AS order_date,
    datediff(current_date(), to_date(order_date))
AS days_since_order,
    CASE
        WHEN datediff(current_date(),
to_date(order_date)) <= 7 THEN '最近一周'
        WHEN datediff(current_date(),
to_date(order_date)) <= 30 THEN '最近一月'
        ELSE '一月前'
    END AS order_period
FROM ecommerce_orders;

-- 3.3 综合报表
SELECT
    region,
    status,
    count(*) AS order_count,
    sum(quantity) AS total_quantity,
    round(sum(quantity * unit_price), 2) AS
total_revenue,
    round(avg(quantity * unit_price), 2) AS
avg_order_value,
    max(quantity * unit_price) AS
max_order_value,
    min(quantity * unit_price) AS min_order_value
FROM ecommerce_orders
GROUP BY region, status
ORDER BY region, total_revenue DESC;
```

案例 2: 员工薪资分析系统

```sql
-- 创建员工薪资表
CREATE TABLE IF NOT EXISTS employee_salary (
    emp_id STRING,
    emp_name STRING,
    department STRING,
    position STRING,
    hire_date STRING,
    base_salary DOUBLE,
    bonus DOUBLE,
    deduction DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
```

```
'/usr/datas/employee_salary.txt'
OVERWRITE INTO TABLE employee_salary;

-- 데이터 확인
SELECT * FROM employee_salary;
SELECT count(*) as total_count FROM
employee_salary;
```

데이터 파일: employee_salary.txt

과제 4: 급여 계산 및 분석

```sql
-- 4.1 실급여 계산 (기본급 + 보너스 - 공제액)
SELECT
    emp_id,
    emp_name,
    department,
    base_salary,
    bonus,
    deduction,
    base_salary + bonus - deduction AS
actual_salary,
    concat('₩', format_number(base_salary + bonus
- deduction, 0)) AS formatted_salary
FROM employee_salary;

-- 4.2 근속연수 계산 및 등급화
SELECT
    emp_id,
    emp_name,
    hire_date,
    datediff(current_date(), hire_date) AS
days_worked,
    round(datediff(current_date(), hire_date) /
365.0, 1) AS years_worked,
    CASE
        WHEN datediff(current_date(), hire_date)
/ 365.0 >= 5 THEN '베테랑'
        WHEN datediff(current_date(), hire_date)
/ 365.0 >= 3 THEN '중견사원'
        WHEN datediff(current_date(), hire_date)
/ 365.0 >= 1 THEN '주니어'
        ELSE '신입사원'
    END AS employee_level
FROM employee_salary
ORDER BY days_worked DESC;

-- 4.3 부서별 급여 통계
SELECT
    department,
    count(*) AS emp_count,
    round(avg(base_salary), 2) AS
avg_base_salary,
    round(avg(bonus), 2) AS avg_bonus,
```

```
'/usr/datas/employee_salary.txt'
OVERWRITE INTO TABLE employee_salary;

-- 验证数据
SELECT * FROM employee_salary;
SELECT count(*) as total_count FROM
employee_salary;
```

数据文件: employee_salary.txt

任务 4: 薪资计算与分析

```sql
-- 4.1 计算实际工资(基本工资 + 奖金 - 扣除)
SELECT
    emp_id,
    emp_name,
    department,
    base_salary,
    bonus,
    deduction,
    base_salary + bonus - deduction AS
actual_salary,
    concat('¥', round(base_salary + bonus -
deduction, 2)) AS formatted_salary
FROM employee_salary;

-- 4.2 计算工龄并分级
SELECT
    emp_id,
    emp_name,
    hire_date,
    datediff(current_date(), hire_date) AS
days_worked,
    round(datediff(current_date(), hire_date) /
365.0, 1) AS years_worked,
    CASE
        WHEN datediff(current_date(), hire_date)
/ 365.0 >= 5 THEN '资深员工'
        WHEN datediff(current_date(), hire_date)
/ 365.0 >= 3 THEN '中级员工'
        WHEN datediff(current_date(), hire_date)
/ 365.0 >= 1 THEN '初级员工'
        ELSE '新员工'
    END AS employee_level
FROM employee_salary
ORDER BY days_worked DESC;

-- 4.3 部门薪资统计
SELECT
    department,
    count(*) AS emp_count,
    round(avg(base_salary), 2) AS
avg_base_salary,
    round(avg(bonus), 2) AS avg_bonus,
```

```
    round(avg(base_salary + bonus - deduction),
2) AS avg_actual_salary,
    round(sum(base_salary + bonus - deduction),
2) AS total_payroll
FROM employee_salary
GROUP BY department
ORDER BY total_payroll DESC;

-- 4.4 급여 등급 구분
SELECT
    emp_name,
    department,
    base_salary + bonus - deduction AS
actual_salary,
    CASE
        WHEN base_salary + bonus - deduction >=
6000000 THEN 'A 등급(>600 만)'
        WHEN base_salary + bonus - deduction >=
4500000 THEN 'B 등급(450-600 만)'
        WHEN base_salary + bonus - deduction >=
3000000 THEN 'C 등급(300-450 만)'
        ELSE 'D 등급(<300 만)'
    END AS salary_grade,
    round((base_salary + bonus - deduction -
1500000) * 0.03, 2) AS estimated_tax
FROM employee_salary
ORDER BY actual_salary DESC;
```

제 10 부: 윈도우 함수 실습(추가 도전) (30 분)

연습 10.1: 판매 순위 테이블 생성

```sql
-- 판매원 실적 테이블 생성
CREATE TABLE IF NOT EXISTS sales_performance (
    sale_date STRING,
    salesperson STRING,
    region STRING,
    sales_amount DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 로컬 파일에서 데이터 로드
LOAD DATA LOCAL INPATH
'/usr/datas/sales_performance.txt'
OVERWRITE INTO TABLE sales_performance;

-- 데이터 확인
SELECT * FROM sales_performance;
SELECT count(*) as total_count FROM
sales_performance;
```

```
    round(avg(base_salary + bonus - deduction),
2) AS avg_actual_salary,
    round(sum(base_salary + bonus - deduction),
2) AS total_payroll
FROM employee_salary
GROUP BY department
ORDER BY total_payroll DESC;

-- 4.4 薪资等级划分
SELECT
    emp_name,
    department,
    base_salary + bonus - deduction AS
actual_salary,
    CASE
        WHEN base_salary + bonus - deduction >=
20000 THEN 'A 级(>20K)'
        WHEN base_salary + bonus - deduction >=
15000 THEN 'B 级(15-20K)'
        WHEN base_salary + bonus - deduction >=
10000 THEN 'C 级(10-15K)'
        ELSE 'D 级(<10K)'
    END AS salary_grade,
    round((base_salary + bonus - deduction -
5000) * 0.03, 2) AS estimated_tax
FROM employee_salary
ORDER BY actual_salary DESC;
```

第十部分: 窗口函数实习(额外挑战) (30 分钟)

练习 10.1: 创建销售排名表

```sql
-- 创建销售员业绩表
CREATE TABLE IF NOT EXISTS sales_performance (
    sale_date STRING,
    salesperson STRING,
    region STRING,
    sales_amount DOUBLE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- 从本地文件加载数据
LOAD DATA LOCAL INPATH
'/usr/datas/sales_performance.txt'
OVERWRITE INTO TABLE sales_performance;

-- 验证数据
SELECT * FROM sales_performance;
SELECT count(*) as total_count FROM
sales_performance;
```

```

**데이터 파일**: sales_performance.txt

### 연습 10.2: 순위 함수

```sql
-- 과제 1: 전체 순위
SELECT
    sale_date,
    salesperson,
    region,
    sales_amount,
    row_number() OVER (ORDER BY sales_amount
DESC) AS row_num,
    rank() OVER (ORDER BY sales_amount DESC) AS
rank,
    dense_rank() OVER (ORDER BY sales_amount
DESC) AS dense_rank
FROM sales_performance;

-- 과제 2: 파티션별 순위 (지역별)
SELECT
    sale_date,
    salesperson,
    region,
    sales_amount,
    rank() OVER (PARTITION BY region ORDER BY
sales_amount DESC) AS region_rank
FROM sales_performance;

-- 과제 3: 월별, 지역별 순위
SELECT
    sale_date,
    salesperson,
    region,
    sales_amount,
    rank() OVER (PARTITION BY sale_date, region
ORDER BY sales_amount DESC) AS
monthly_region_rank
FROM sales_performance;
```

실습 정리 및 평가

자가 평가 체크리스트

모든 연습을 완료한 후 다음 질문에 답해보세요:

1. **문자열 함수 부분**
   - concat, substr, length 등의 함수를 능숙하게


```

**数据文件**: sales_performance.txt

### 练习 10.2: 排名函数

```sql
-- 任务 1: 全局排名
SELECT
    sale_date,
    salesperson,
    region,
    sales_amount,
    row_number() OVER (ORDER BY sales_amount
DESC) AS row_num,
    rank() OVER (ORDER BY sales_amount DESC) AS
rank,
    dense_rank() OVER (ORDER BY sales_amount
DESC) AS dense_rank
FROM sales_performance;

-- 任务 2: 分区排名(按地区)
SELECT
    sale_date,
    salesperson,
    region,
    sales_amount,
    rank() OVER (PARTITION BY region ORDER BY
sales_amount DESC) AS region_rank
FROM sales_performance;

-- 任务 3: 按月份和地区排名
SELECT
    sale_date,
    salesperson,
    region,
    sales_amount,
    rank() OVER (PARTITION BY sale_date, region
ORDER BY sales_amount DESC) AS
monthly_region_rank
FROM sales_performance;
```

实习总结与评估

自我评估清单

请在完成所有练习后，回答以下问题：

1. **字符串函数部分**
   - 你能熟练使用 concat、substr、length 等函数

사용할 수 있나요?
  - 정규표현식 치환 사용법을 이해하나요?

2. **날짜 함수 부분**
  - 두 날짜 사이의 일수 차이를 정확히 계산할 수 있나요?
  - date_add 와 date_sub 를 사용한 날짜 계산을 할 수 있나요?

3. **수학 함수 부분**
  - round, floor, ceil 의 차이를 이해하나요?
  - 난수 함수를 올바르게 사용할 수 있나요?

4. **조건 함수 부분**
  - IF 와 CASE WHEN 을 능숙하게 사용할 수 있나요?
  - NULL 값을 올바르게 처리할 수 있나요?

5. **집합 함수 부분**
  - Array 와 Map 연산을 이해하나요?
  - collect_set 과 collect_list 를 사용할 수 있나요?

6. **집계 함수 부분**
  - GROUP BY 를 사용한 그룹별 통계를 올바르게 수행할 수 있나요?
  - 다양한 집계 함수의 용도를 이해하나요?

7. **데이터 마스킹 부분**
  - 다양한 유형의 민감 데이터에 대한 마스킹 방안을 설계할 수 있나요?
  - 데이터 보안의 중요성을 이해하나요?

도전 과제

```sql
-- 종합 도전: 완전한 데이터 분석 리포트 작성
-- 이번 실습에서 배운 모든 함수 유형을 결합

CREATE TABLE IF NOT EXISTS comprehensive_report
```

吗？
  - 你理解正则表达式替换的用法吗？

2. **日期函数部分**
  - 你能正确计算两个日期之间的天数差吗？
  - 你会使用 date_add 和 date_sub 进行日期计算吗？

3. **数学函数部分**
  - 你理解 round、floor、ceil 的区别吗？
  - 你能正确使用随机数函数吗？

4. **条件函数部分**
  - 你能熟练使用 IF 和 CASE WHEN 吗？
  - 你会正确处理 NULL 值吗？

5. **集合函数部分**
  - 你理解 Array 和 Map 的操作吗？
  - 你会使用 collect_set 和 collect_list 吗？

6. **聚合函数部分**
  - 你能正确使用 GROUP BY 进行分组统计吗？
  - 你理解各种聚合函数的用途吗？

7. **数据脱敏部分**
  - 你能为不同类型的敏感数据设计脱敏方案吗？
  - 你理解数据安全的重要性吗?

挑战题

```sql
-- 综合挑战：创建一个完整的数据分析报表
-- 结合本次实习学到的所有函数类型

CREATE TABLE IF NOT EXISTS comprehensive_report
```

```
AS
SELECT
    -- 문자열 처리
    upper(substr(emp_name, 1, 1)) AS first_letter,

    -- 날짜 계산
    datediff(current_date(), hire_date) AS days_employed,

    -- 수학 계산
    round(base_salary * 1.1, 2) AS projected_salary,

    -- 조건 판단
    CASE
        WHEN base_salary > 4500000 THEN '고급'
        WHEN base_salary > 3000000 THEN '중급'
        ELSE '초급'
    END AS salary_level,

    -- 집계 정보
    department,
    count(*) OVER (PARTITION BY department) AS dept_count

FROM employee_salary;

-- 결과 조회
SELECT * FROM comprehensive_report;
```

정리 작업

```sql
-- 실습 데이터를 정리하려면 다음 명령을 실행하세요:
-- DROP TABLE IF EXISTS employees;
-- DROP TABLE IF EXISTS orders;
-- DROP TABLE IF EXISTS product_sales;
-- DROP TABLE IF EXISTS student_scores;
-- DROP TABLE IF EXISTS user_interests;
-- DROP TABLE IF EXISTS sales_detail;
-- DROP TABLE IF EXISTS customer_info;
-- DROP TABLE IF EXISTS type_conversion_test;
-- DROP TABLE IF EXISTS ecommerce_orders;
-- DROP TABLE IF EXISTS employee_salary;
-- DROP TABLE IF EXISTS sales_performance;
-- DROP TABLE IF EXISTS comprehensive_report;
```

축하합니다!


```
AS
SELECT
    -- 字符串处理
    upper(substr(emp_name, 1, 1)) AS first_letter,

    -- 日期计算
    datediff(current_date(), hire_date) AS days_employed,

    -- 数学计算
    round(base_salary * 1.1, 2) AS projected_salary,

    -- 条件判断
    CASE
        WHEN base_salary > 15000 THEN '高薪'
        WHEN base_salary > 10000 THEN '中薪'
        ELSE '基础'
    END AS salary_level,

    -- 聚合信息
    department,
    count(*) OVER (PARTITION BY department) AS dept_count

FROM employee_salary;

-- 查看结果
SELECT * FROM comprehensive_report;
```

退出清理

```sql
-- 如需清理实习数据,可以执行以下命令:
-- DROP TABLE IF EXISTS employees;
-- DROP TABLE IF EXISTS orders;
-- DROP TABLE IF EXISTS product_sales;
-- DROP TABLE IF EXISTS student_scores;
-- DROP TABLE IF EXISTS user_interests;
-- DROP TABLE IF EXISTS sales_detail;
-- DROP TABLE IF EXISTS customer_info;
-- DROP TABLE IF EXISTS type_conversion_test;
-- DROP TABLE IF EXISTS ecommerce_orders;
-- DROP TABLE IF EXISTS employee_salary;
-- DROP TABLE IF EXISTS sales_performance;
-- DROP TABLE IF EXISTS comprehensive_report;
```

恭喜完成!

Hive 함수 종합 실습을 완료하셨습니다. 이번 실습을 통해 다음을 습득하셨습니다:

- ✅ 문자열 함수의 유연한 활용
- ✅ 날짜 시간 함수의 계산 기법
- ✅ 수학 함수의 실제 응용
- ✅ 조건 함수의 논리 판단
- ✅ 집합 함수의 복합 타입 처리
- ✅ 집계 함수의 데이터 통계
- ✅ 데이터 마스킹의 보안 처리
- ✅ 타입 변환의 올바른 사용

계속 학습하고 끊임없이 실습하세요!

你已经完成了 Hive 函数的综合实习。通过本次实习，你应该已经掌握了:

- ✅ 字符串函数的灵活运用
- ✅ 日期时间函数的计算技巧
- ✅ 数学函数的实际应用
- ✅ 条件函数的逻辑判断
- ✅ 集合函数的复杂类型处理
- ✅ 聚合函数的数据统计
- ✅ 数据脱敏的安全处理
- ✅ 类型转换的正确使用

继续保持学习，不断实践！