Apache Hive 第 7 章 函数综合实践工作手册

> 课程信息 - 课程名称：大数据技术 —— Apache Hive - 章节：第7章 Hive 内置函数综合实践 - 实践时长：约 4 小时（240 分钟） - 数据库名称：`unit7db` - 数据集目录：`/usr/datas/unit7/` - Beeline 启动命令：`hive-bee`

目录

1. [实践环境准备](#1-实践环境准备)
2. [数据库与数据表创建](#2-数据库与数据表创建)
3. [数据集加载](#3-数据集加载)
4. [字符串函数实践（String Functions）](#4-字符串函数实践)
5. [数值与数学函数实践（Math Functions）](#5-数值与数学函数实践)
6. [日期与时间函数实践（Date/Time Functions）](#6-日期与时间函数实践)
7. [条件判断函数实践（Conditional Functions）](#7-条件判断函数实践)
8. [聚合函数实践（Aggregate Functions）](#8-聚合函数实践)
9. [窗口函数实践（Window Functions）](#9-窗口函数实践)
10. [复杂类型函数实践（Complex Type Functions）](#10-复杂类型函数实践)
11. [类型转换函数实践（Type Conversion）](#11-类型转换函数实践)
12. [综合练习题](#12-综合练习题)
13. [课堂总结与评分标准](#13-课堂总结与评分标准)

---

1. 实践环境准备

1.1 目标

- 确认 Hive 与 Beeline 正常运行

- 确认数据集文件存在

- 创建并切换到实践数据库

1.2 时间分配：15 分钟

1.3 启动 Beeline

打开终端，输入以下命令启动 Beeline：

[BASH]

```
hive-bee
```

启动成功后，会看到如下提示符：

```
Beeline version x.x.x by Apache Hive
beeline>
```

连接 HiveServer2（如未自动连接，执行以下命令）：

[SQL]

```
!connect jdbc:hive2://localhost:10000
-- 用户名：hiveuser
-- 密码：123456
```

1.4 确认数据集文件

在新终端窗口中执行（不关闭 Beeline）：

[BASH]

```
ls -l /usr/datas/unit7/
```

应该可以看到如下文件（或类似文件）：

```
employees.txt
departments.txt
orders.txt
order_items.txt
products.txt
students.txt
```

> 提示： 如果目录不存在，请联系教师。

1.5 检查当前数据库列表

在 Beeline 中执行：

[SQL]

```
SHOW DATABASES;
```

1.6 创建实践数据库

[SQL]

```
CREATE DATABASE IF NOT EXISTS unit7db
LOCATION '/user/hive/warehouse/unit7db'
COMMENT 'Unit 7 Hive Functions Practice Database';
```

切换到实践数据库：

[SQL]

```
USE unit7db;
```

确认当前数据库：

[SQL]

```
SELECT current_database();
```

> ✅ 检查点 1： 确认已切换到 `unit7db`，再进行下一步。

---

2. 数据库与数据表创建

2.1 目标

- 根据业务场景设计并创建数据表

- 理解 Hive 表结构定义语法

2.2 时间分配：20 分钟

2.3 创建员工表（employees）

[SQL]

```
CREATE TABLE IF NOT EXISTS unit7db.employees (
    emp_id      INT         COMMENT '员工编号',
```

```
    emp_name    STRING      COMMENT '员工姓名',
    gender      STRING      COMMENT '性别 M/F',
    birth_date  STRING      COMMENT '出生日期 yyyy-MM-dd',
    hire_date   STRING      COMMENT '入职日期 yyyy-MM-dd',
    dept_id     INT         COMMENT '部门编号',
    job_title   STRING      COMMENT '职位',
    salary      DOUBLE      COMMENT '月薪（元）',
    email       STRING      COMMENT '邮箱地址',
    city        STRING      COMMENT '所在城市'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ('skip.header.line.count'='1');
```

2.4 创建部门表（departments）

[SQL]

```
CREATE TABLE IF NOT EXISTS unit7db.departments (
    dept_id     INT     COMMENT '部门编号',
    dept_name   STRING  COMMENT '部门名称',
    location    STRING  COMMENT '办公地点',
    manager_id  INT     COMMENT '部门经理编号'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ('skip.header.line.count'='1');
```

2.5 创建订单表（orders）

[SQL]

```
CREATE TABLE IF NOT EXISTS unit7db.orders (
    order_id        INT     COMMENT '订单编号',
    customer_name   STRING  COMMENT '客户姓名',
    order_date      STRING  COMMENT '订单日期 yyyy-MM-dd',
    total_amount    DOUBLE  COMMENT '订单总金额',
    status          STRING  COMMENT '订单状态',
    city            STRING  COMMENT '城市'
)
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ('skip.header.line.count'='1');
```

2.6 创建订单明细表（order_items）

[SQL]

```
CREATE TABLE IF NOT EXISTS unit7db.order_items (
    item_id     INT     COMMENT '明细编号',
    order_id    INT     COMMENT '订单编号',
    product_id  INT     COMMENT '产品编号',
    quantity    INT     COMMENT '数量',
    unit_price  DOUBLE  COMMENT '单价',
    discount    DOUBLE  COMMENT '折扣率'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ('skip.header.line.count'='1');
```

2.7 创建产品表（products）

[SQL]

```
CREATE TABLE IF NOT EXISTS unit7db.products (
    product_id      INT     COMMENT '产品编号',
    product_name    STRING  COMMENT '产品名称',
    category        STRING  COMMENT '产品类别',
    unit_price      DOUBLE  COMMENT '单价',
    stock_qty       INT     COMMENT '库存数量',
    supplier        STRING  COMMENT '供应商'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ('skip.header.line.count'='1');
```

2.8 创建学生成绩表（students）

[SQL]

```
CREATE TABLE IF NOT EXISTS unit7db.students (
    stu_id      INT     COMMENT '学生编号',
```

```
    stu_name    STRING  COMMENT '学生姓名',
    class_name  STRING  COMMENT '班级',
    math_score  INT     COMMENT '数学成绩',
    eng_score   INT     COMMENT '英语成绩',
    sci_score   INT     COMMENT '科学成绩',
    exam_date   STRING  COMMENT '考试日期'
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ('skip.header.line.count'='1');
```

2.9 确认所有表已创建

[SQL]

```
SHOW TABLES;
```

> ✅ 检查点 2： 确认显示出 6 张表，再进行下一步。

---

3. 数据集加载

3.1 目标

- 使用 `LOAD DATA LOCAL INPATH` 命令加载数据

- 验证数据加载成功

3.2 时间分配：15 分钟

3.3 准备数据文件

在新终端中执行以下命令，创建数据集文件：

[BASH]

```
# 创建目录
sudo mkdir -p /usr/datas/unit7

# 创建员工数据文件
cat > /usr/datas/unit7/employees.txt << 'EOF'
emp_id,emp_name,gender,birth_date,hire_date,dept_id,job_title,salary,email,city
```

```
1001,张伟,M,1985-03-15,2010-06-01,10,软件工程师,15000,zhangwei@company.com,北京
1002,李娜,F,1990-07-22,2015-09-01,20,数据分析师,12000,lina@company.com,上海
1003,王强,M,1988-11-05,2012-03-15,10,高级工程师,20000,wangqiang@company.com,北京
1004,赵雪,F,1992-01-30,2018-07-01,30,人力资源专员,9000,zhaoxue@company.com,广州
1005,陈明,M,1986-05-18,2011-04-01,20,数据科学家,25000,chenming@company.com,上海
1006,刘芳,F,1994-09-12,2020-01-15,10,初级工程师,8000,liufang@company.com,北京
1007,杨洋,M,1983-12-25,2008-08-01,40,财务主管,18000,yangyang@company.com,深圳
1008,周丽,F,1991-04-08,2016-11-01,30,招聘专员,9500,zhouli@company.com,广州
1009,吴刚,M,1987-08-20,2013-02-01,40,财务分析师,14000,wugang@company.com,深圳
1010,孙雨,F,1995-06-14,2021-03-01,20,数据工程师,11000,sunyu@company.com,上海
1011,马超,M,1989-02-28,2014-05-01,10,全栈开发,17000,machao@company.com,北京
1012,朱婷,F,1993-10-16,2019-08-01,30,培训专员,8500,zhuting@company.com,广州
1013,胡军,M,1984-07-03,2009-01-01,40,首席财务官,35000,hujun@company.com,深圳
1014,林晓,F,1996-03-27,2022-06-01,20,数据实习生,6000,linxiao@company.com,上海
1015,黄磊,M,1982-09-09,2007-10-01,10,技术总监,40000,huanglei@company.com,北京
EOF

# 创建部门数据文件
cat > /usr/datas/unit7/departments.txt << 'EOF'
dept_id,dept_name,location,manager_id
10,技术部,北京,1015
20,数据部,上海,1005
30,人力资源部,广州,1004
40,财务部,深圳,1013
EOF

# 创建订单数据文件
cat > /usr/datas/unit7/orders.txt << 'EOF'
order_id,customer_name,order_date,total_amount,status,city
2001,张三,2024-01-05,1580.50,已完成,北京
2002,李四,2024-01-08,320.00,已完成,上海
2003,王五,2024-01-12,4500.00,已发货,广州
2004,赵六,2024-01-15,899.99,已完成,深圳
2005,孙七,2024-01-20,12000.00,处理中,北京
2006,周八,2024-02-02,250.50,已完成,上海
2007,吴九,2024-02-10,3300.00,已完成,广州
2008,郑十,2024-02-14,780.00,已取消,北京
2009,冯一,2024-02-18,5600.00,已发货,深圳
2010,陈二,2024-02-22,1200.00,处理中,上海
2011,楚三,2024-03-01,8800.00,已完成,北京
2012,秦四,2024-03-05,450.00,已完成,广州
2013,许五,2024-03-10,2100.00,已发货,深圳
2014,韩六,2024-03-15,680.00,已取消,上海
2015,曹七,2024-03-20,9500.00,已完成,北京
```

```
EOF

# 创建订单明细数据文件
cat > /usr/datas/unit7/order_items.txt << 'EOF'
item_id,order_id,product_id,quantity,unit_price,discount
1,2001,101,2,500.00,0.10
2,2001,102,3,200.00,0.05
3,2002,103,1,320.00,0.00
4,2003,104,5,800.00,0.10
5,2003,101,1,500.00,0.10
6,2004,102,3,200.00,0.15
7,2005,105,2,5500.00,0.05
8,2005,101,2,500.00,0.00
9,2006,103,1,250.00,0.00
10,2007,104,4,800.00,0.05
11,2008,102,3,200.00,0.15
12,2009,105,1,5500.00,0.02
13,2010,101,2,500.00,0.10
14,2011,105,1,5500.00,0.05
15,2012,103,2,250.00,0.10
EOF

# 创建产品数据文件
cat > /usr/datas/unit7/products.txt << 'EOF'
product_id,product_name,category,unit_price,stock_qty,supplier
101,笔记本电脑A,电子产品,5000.00,120,联想科技
102,无线鼠标,外设,200.00,500,罗技
103,机械键盘,外设,320.00,300,达尔优
104,4K显示器,外设,800.00,80,三星
105,游戏本B,电子产品,8500.00,50,华硕
106,USB集线器,外设,150.00,400,绿联
107,移动硬盘,存储,450.00,200,西数
108,固态硬盘,存储,600.00,150,三星
EOF

# 创建学生成绩数据文件
cat > /usr/datas/unit7/students.txt << 'EOF'
stu_id,stu_name,class_name,math_score,eng_score,sci_score,exam_date
3001,张小明,大数据2201,92,85,88,2024-01-15
3002,李小红,大数据2201,78,90,76,2024-01-15
3003,王大壮,大数据2202,65,72,80,2024-01-15
3004,赵小雅,大数据2202,95,88,92,2024-01-15
3005,孙晓光,大数据2201,55,60,58,2024-01-15
3006,周佳佳,大数据2203,88,92,85,2024-01-15
3007,吴小波,大数据2203,73,68,77,2024-01-15
3008,郑雅丽,大数据2202,90,95,91,2024-01-15
```

```
3009,冯建国,大数据2201,82,75,84,2024-01-15
3010,陈晓丹,大数据2203,60,65,70,2024-01-15
3011,楚天舒,大数据2202,45,55,50,2024-01-15
3012,秦小强,大数据2201,98,96,99,2024-01-15
EOF

echo "数据文件创建完成！"
ls -la /usr/datas/unit7/
```

3.4 在 Beeline 中加载数据

切换回 Beeline 窗口，依次执行：

[SQL]

```
-- 加载员工数据
LOAD DATA LOCAL INPATH '/usr/datas/unit7/employees.txt'
OVERWRITE INTO TABLE unit7db.employees;

-- 加载部门数据
LOAD DATA LOCAL INPATH '/usr/datas/unit7/departments.txt'
OVERWRITE INTO TABLE unit7db.departments;

-- 加载订单数据
LOAD DATA LOCAL INPATH '/usr/datas/unit7/orders.txt'
OVERWRITE INTO TABLE unit7db.orders;

-- 加载订单明细数据
LOAD DATA LOCAL INPATH '/usr/datas/unit7/order_items.txt'
OVERWRITE INTO TABLE unit7db.order_items;

-- 加载产品数据
LOAD DATA LOCAL INPATH '/usr/datas/unit7/products.txt'
OVERWRITE INTO TABLE unit7db.products;

-- 加载学生成绩数据
LOAD DATA LOCAL INPATH '/usr/datas/unit7/students.txt'
OVERWRITE INTO TABLE unit7db.students;
```

3.5 验证数据加载

[SQL]

```
-- 查看各表记录数
SELECT COUNT(*) AS emp_count FROM unit7db.employees;
```

```
SELECT COUNT(*) AS dept_count FROM unit7db.departments;
SELECT COUNT(*) AS order_count FROM unit7db.orders;
SELECT COUNT(*) AS item_count FROM unit7db.order_items;
SELECT COUNT(*) AS product_count FROM unit7db.products;
SELECT COUNT(*) AS stu_count FROM unit7db.students;
```

预期结果：

表名	记录数
employees	15
departments	4
orders	15
order_items	15
products	8
students	12

[SQL]

```
-- 预览各表前3条数据
SELECT * FROM unit7db.employees LIMIT 3;
SELECT * FROM unit7db.departments LIMIT 3;
SELECT * FROM unit7db.orders LIMIT 3;
```

> ✅ 检查点 3： 确认所有表数据加载正确后，进入函数实践环节。

---

4. 字符串函数实践

4.1 目标

- 掌握 Hive 常用字符串处理函数

- 能在实际业务场景中灵活运用字符串函数

4.2 时间分配：25 分钟

4.3 函数总览

函数	说明

`LENGTH(str)`	返回字符串长度
`UPPER(str)` / `LOWER(str)`	大/小写转换
`TRIM(str)` / `LTRIM(str)` / `RTRIM(str)`	去除空格
`SUBSTR(str, pos, len)`	截取子串
`CONCAT(str1, str2, ...)`	字符串拼接
`CONCAT_WS(sep, str1, str2, ...)`	带分隔符拼接
`REPLACE(str, old, new)`	字符串替换
`INSTR(str, substr)`	查找子串位置
`SPLIT(str, pat)`	按分隔符分割
`REGEXP_REPLACE(str, pat, rep)`	正则替换
`LPAD(str, len, pad)` / `RPAD(str, len, pad)`	填充对齐
`REVERSE(str)`	字符串反转
`TRANSLATE(str, from, to)`	字符替换

4.4 实践步骤

步骤 1：基础字符串操作

[SQL]

```
-- 【示例 4-1】查看员工姓名长度与大小写转换
SELECT
    emp_name,
    LENGTH(emp_name)            AS name_length,
    UPPER(emp_name)             AS name_upper,
    LOWER(emp_name)             AS name_lower
FROM unit7db.employees
LIMIT 5;
```

[SQL]

```
-- 【示例 4-2】邮箱大小写统一处理
SELECT
    emp_id,
    emp_name,
    email,
    LOWER(TRIM(email))          AS clean_email,
```

```
    UPPER(SUBSTR(email, 1, INSTR(email,'@')-1)) AS email_prefix_upper
FROM unit7db.employees
LIMIT 5;
```

步骤 2：字符串截取与拼接

[SQL]

```
-- 【示例 4-3】截取出生年份并拼接完整信息
SELECT
    emp_name,
    birth_date,
    SUBSTR(birth_date, 1, 4)    AS birth_year,
    SUBSTR(birth_date, 6, 2)    AS birth_month,
    SUBSTR(birth_date, 9, 2)    AS birth_day,
    CONCAT('员工：', emp_name, ' | 出生年份：', SUBSTR(birth_date, 1, 4)) AS emp_info
FROM unit7db.employees
LIMIT 5;
```

[SQL]

```
-- 【示例 4-4】使用 CONCAT_WS 构建报告字段
SELECT
    CONCAT_WS(' | ',
        emp_name,
        job_title,
        city,
        CONCAT('薪资：', CAST(salary AS STRING), '元')
    ) AS employee_report
FROM unit7db.employees
LIMIT 8;
```

步骤 3：字符串查找与替换

[SQL]

```
-- 【示例 4-5】查找邮箱中 @ 符号的位置
SELECT
    emp_name,
    email,
    INSTR(email, '@')           AS at_position,
    SUBSTR(email, INSTR(email, '@')+1) AS email_domain
FROM unit7db.employees;
```

[SQL]

```sql
-- 【示例 4-6】替换邮箱域名
SELECT
    emp_name,
    email,
    REPLACE(email, 'company.com', 'newdomain.cn') AS new_email
FROM unit7db.employees
LIMIT 5;
```

步骤 4：正则表达式与格式化

[SQL]

```sql
-- 【示例 4-7】使用 LPAD 统一员工编号格式（6位）
SELECT
    emp_id,
    LPAD(CAST(emp_id AS STRING), 6, '0') AS formatted_emp_id,
    emp_name
FROM unit7db.employees
LIMIT 5;
```

[SQL]

```sql
-- 【示例 4-8】REGEXP_REPLACE 去除邮箱中的数字
SELECT
    emp_name,
    email,
    REGEXP_REPLACE(email, '[0-9]', '') AS email_no_numbers
FROM unit7db.employees
LIMIT 5;
```

4.5 练习题（字符串函数）

【练习 4-A】 查询所有员工的邮箱地址，提取邮箱用户名部分（@ 之前的部分），并将其转为全大写。输出列名为：`emp_name`、`email`、`email_username`。

> 提示：使用 `SUBSTR`、`INSTR`、`UPPER` 组合实现。

[SQL]

```sql
-- 在此处编写你的SQL语句：
```

【练习 4-B】 查询所有员工，将 `emp_name`（员工姓名）、`job_title`（职位）和 `city`（城市）用 " - " 作为分隔符拼接为一个字段 `profile`。同时计算 `profile` 的总字符长度。

[SQL]

```
-- 在此处编写你的SQL语句：
```

【练习 4-C】 查询所有产品，将产品名称中的"A"替换为"Pro"，将"B"替换为"Max"，输出原名称和修改后名称。

[SQL]

```
-- 在此处编写你的SQL语句：
```

5. 数值与数学函数实践

5.1 目标

- 掌握 Hive 数值计算与取整函数

- 能够处理财务类数据的精度问题

5.2 时间分配：20 分钟

5.3 函数总览

函数	说明
`ROUND(x, d)`	四舍五入，d 为小数位
`FLOOR(x)`	向下取整

`CEIL(x)` / `CEILING(x)`	向上取整
`ABS(x)`	绝对值
`SQRT(x)`	平方根
`POW(x, y)` / `POWER(x, y)`	x 的 y 次方
`MOD(x, y)`	取余
`RAND()`	生成 0~1 随机数
`LOG(base, x)`	对数运算
`EXP(x)`	e 的 x 次方
`SIGN(x)`	返回符号（-1/0/1）
`GREATEST(v1,v2,...)`	最大值
`LEAST(v1,v2,...)`	最小值

5.4 实践步骤

步骤 1：薪资计算

[SQL]

```
-- 【示例 5-1】薪资取整比较
SELECT
    emp_name,
    salary,
    ROUND(salary, -3)          AS salary_rounded_k,   -- 精确到千位
    FLOOR(salary / 1000)       AS salary_floor_k,      -- 薪资（千元，向下取整）
    CEIL(salary / 1000)        AS salary_ceil_k        -- 薪资（千元，向上取整）
FROM unit7db.employees
ORDER BY salary DESC
LIMIT 8;
```

[SQL]

```
-- 【示例 5-2】月薪、年薪、税后计算
SELECT
    emp_name,
    salary                     AS monthly_salary,
    salary * 12                AS annual_salary,
    ROUND(salary * 0.85, 2)    AS after_tax_salary,    -- 假设税率15%
    ROUND(ABS(salary - 15000) / 15000 * 100, 2) AS diff_from_avg_pct  -- 与均值差距百分比
```

```
FROM unit7db.employees
ORDER BY salary DESC;
```

步骤 2：价格与折扣计算

[SQL]

```
-- 【示例 5-3】计算订单实际金额
SELECT
    oi.order_id,
    p.product_name,
    oi.quantity,
    oi.unit_price,
    oi.discount,
    ROUND(oi.quantity * oi.unit_price * (1 - oi.discount), 2) AS actual_amount,
    ROUND(oi.quantity * oi.unit_price * oi.discount, 2)        AS discount_amount
FROM unit7db.order_items oi
JOIN unit7db.products p ON oi.product_id = p.product_id
LIMIT 10;
```

步骤 3：数学函数综合

[SQL]

```
-- 【示例 5-4】MOD 取余 — 区分奇偶员工编号
SELECT
    emp_id,
    emp_name,
    MOD(emp_id, 2)              AS is_even,   -- 0=偶数, 1=奇数
    CASE WHEN MOD(emp_id, 2) = 0 THEN '偶数' ELSE '奇数' END AS even_odd
FROM unit7db.employees
ORDER BY emp_id;
```

[SQL]

```
-- 【示例 5-5】GREATEST / LEAST 找三科最高最低分
SELECT
    stu_name,
    math_score, eng_score, sci_score,
    GREATEST(math_score, eng_score, sci_score) AS highest_score,
    LEAST(math_score, eng_score, sci_score)    AS lowest_score,
    ROUND((math_score + eng_score + sci_score) / 3.0, 1) AS avg_score
FROM unit7db.students;
```

5.5 练习题（数值函数）

【练习 5-A】 查询订单明细表，计算每条明细的含折扣后的实付金额（quantity × unit_price × (1 - discount)），结果保留2位小数，并按实付金额降序排列，显示前8条。

[SQL]

```
-- 在此处编写你的SQL语句：
```

【练习 5-B】 查询员工表，使用 `ROUND`、`FLOOR`、`CEIL` 分别对 `salary / 1000`（薪资除以1000）进行取整，显示员工姓名、原始薪资及三种取整结果，输出列名自行命名。

[SQL]

```
-- 在此处编写你的SQL语句：
```

6. 日期与时间函数实践

6.1 目标

- 掌握 Hive 日期处理函数

- 能够计算日期差、格式化日期

6.2 时间分配：25 分钟

6.3 函数总览

函数	说明
`CURRENT_DATE`	当前日期
`CURRENT_TIMESTAMP`	当前时间戳

`TO_DATE(str)`	字符串转日期
`YEAR(date)` / `MONTH(date)` / `DAY(date)`	提取年/月/日
`HOUR(ts)` / `MINUTE(ts)` / `SECOND(ts)`	提取时/分/秒
`DATE_ADD(date, days)`	日期加天数
`DATE_SUB(date, days)`	日期减天数
`DATEDIFF(d1, d2)`	两日期相差天数
`FROM_UNIXTIME(unix_time, fmt)`	Unix时间戳转日期
`UNIX_TIMESTAMP(date, fmt)`	日期转Unix时间戳
`DATE_FORMAT(date, fmt)`	日期格式化
`WEEKOFYEAR(date)`	返回日期所在周数
`DAYOFWEEK(date)`	返回星期几（1=周日）
`TRUNC(date, fmt)`	日期截断（月/年初）
`MONTHS_BETWEEN(d1, d2)`	两日期相差月数
`LAST_DAY(date)`	当月最后一天

6.4 实践步骤

步骤 1：基础日期函数

[SQL]

```
-- 【示例 6-1】提取日期各部分
SELECT
    CURRENT_DATE                AS today,
    CURRENT_TIMESTAMP           AS now,
    YEAR(CURRENT_DATE)          AS current_year,
    MONTH(CURRENT_DATE)         AS current_month,
    DAY(CURRENT_DATE)           AS current_day,
    WEEKOFYEAR(CURRENT_DATE)    AS week_of_year;
```

[SQL]

```
-- 【示例 6-2】提取员工入职年月日
SELECT
    emp_name,
    hire_date,
    YEAR(hire_date)             AS hire_year,
```

```
    MONTH(hire_date)            AS hire_month,
    DAY(hire_date)              AS hire_day,
    DATE_FORMAT(hire_date, 'yyyy年MM月dd日') AS hire_date_cn
FROM unit7db.employees
LIMIT 5;
```

步骤 2：日期计算

[SQL]

```
-- 【示例 6-3】计算员工在职天数与在职年数
SELECT
    emp_name,
    hire_date,
    DATEDIFF(CURRENT_DATE, hire_date)               AS days_worked,
    FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365)  AS years_worked,
    DATE_ADD(hire_date, 365)                        AS one_year_anniversary
FROM unit7db.employees
ORDER BY days_worked DESC
LIMIT 8;
```

[SQL]

```
-- 【示例 6-4】订单日期分析
SELECT
    order_id,
    customer_name,
    order_date,
    YEAR(order_date)            AS order_year,
    MONTH(order_date)           AS order_month,
    DAYOFWEEK(order_date)       AS day_of_week,
    DATE_ADD(order_date, 7)     AS delivery_estimate,
    DATEDIFF(CURRENT_DATE, order_date) AS days_since_order
FROM unit7db.orders
ORDER BY order_date;
```

步骤 3：高级日期处理

[SQL]

```
-- 【示例 6-5】计算员工年龄（基于出生日期）
SELECT
    emp_name,
    birth_date,
    FLOOR(DATEDIFF(CURRENT_DATE, birth_date) / 365) AS age,
    LAST_DAY(birth_date)        AS birth_month_last_day,
    TRUNC(birth_date, 'MM')     AS birth_month_first_day
```

```
FROM unit7db.employees
ORDER BY birth_date;
```

[SQL]

```
-- 【示例 6-6】按月份统计订单数
SELECT
    DATE_FORMAT(order_date, 'yyyy-MM') AS order_month,
    COUNT(*)                           AS order_count,
    ROUND(SUM(total_amount), 2)        AS monthly_revenue
FROM unit7db.orders
GROUP BY DATE_FORMAT(order_date, 'yyyy-MM')
ORDER BY order_month;
```

[SQL]

```
-- 【示例 6-7】UNIX时间戳转换演示
SELECT
    UNIX_TIMESTAMP('2024-01-05', 'yyyy-MM-dd')  AS unix_ts,
    FROM_UNIXTIME(1704384000, 'yyyy-MM-dd HH:mm:ss') AS from_unix;
```

6.5 练习题（日期函数）

【练习 6-A】 查询所有员工，计算其在职年数（`FLOOR(DATEDIFF/365)`），并按在职年数降序排列，显示：`emp_name`、`hire_date`、`years_worked`（在职年数）、`next_anniversary`（下一个入职周年纪念日，即 `hire_date` 加上 `(years_worked+1)*365` 天）。

[SQL]

```
-- 在此处编写你的SQL语句：
```

【练习 6-B】 统计 2024年每月订单总金额和订单数量，结果按月份排序，输出列：`order_month`、`order_count`、`total_revenue`（保留2位小数）。

[SQL]

```
-- 在此处编写你的SQL语句：
```

---

7. 条件判断函数实践

7.1 目标

- 掌握 IF、CASE WHEN、COALESCE、NVL 等条件函数

- 能根据业务规则实现数据分类与标签化

7.2 时间分配：20 分钟

7.3 函数总览

函数	说明
`IF(cond, true_val, false_val)`	简单条件判断
`CASE WHEN ... THEN ... ELSE ... END`	多条件分支
`COALESCE(v1, v2, ...)`	返回第一个非NULL值
`NVL(val, default_val)`	NULL替换（同COALESCE两参数版）
`NULLIF(v1, v2)`	若v1=v2则返回NULL
`IFF(cond, true_val, false_val)`	IF 的别名
`DECODE(expr, s1, r1, s2, r2, ...)`	多值匹配

7.4 实践步骤

步骤 1：IF 函数

[SQL]

```
-- 【示例 7-1】根据薪资判断高薪低薪
SELECT
    emp_name,
    salary,
```

```
    IF(salary >= 15000, '高薪', '普通') AS salary_level,
    IF(gender = 'M', '男', '女')        AS gender_cn
FROM unit7db.employees
ORDER BY salary DESC;
```

步骤 2：CASE WHEN 多分支

[SQL]

```
-- 【示例 7-2】薪资分级（5个等级）
SELECT
    emp_name,
    salary,
    CASE
        WHEN salary >= 30000 THEN '🌟 高管级'
        WHEN salary >= 20000 THEN '🟢 高级'
        WHEN salary >= 15000 THEN '🔵 中级'
        WHEN salary >= 10000 THEN '🟡 初级'
        ELSE '🟠 实习/基础'
    END AS salary_grade
FROM unit7db.employees
ORDER BY salary DESC;
```

[SQL]

```
-- 【示例 7-3】订单状态中文说明与优先级
SELECT
    order_id,
    customer_name,
    status,
    CASE status
        WHEN '已完成' THEN '完成'
        WHEN '已发货' THEN '运输中'
        WHEN '处理中' THEN '进行中'
        WHEN '已取消' THEN '取消'
        ELSE '未知'
    END AS status_label,
    CASE status
        WHEN '处理中' THEN 1
        WHEN '已发货' THEN 2
        WHEN '已完成' THEN 3
        WHEN '已取消' THEN 4
        ELSE 5
    END AS priority
FROM unit7db.orders
```

```
ORDER BY priority;
```

步骤 3：学生成绩评级

[SQL]

```
-- 【示例 7-4】成绩等级评定（综合运用 CASE WHEN）
SELECT
    stu_name,
    class_name,
    math_score,
    eng_score,
    sci_score,
    ROUND((math_score + eng_score + sci_score) / 3.0, 1) AS avg_score,
    CASE
        WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 90 THEN 'A（优秀）'
        WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 75 THEN 'B（良好）'
        WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 60 THEN 'C（及格）'
        ELSE 'D（不及格）'
    END AS grade
FROM unit7db.students
ORDER BY avg_score DESC;
```

步骤 4：COALESCE / NVL

[SQL]

```
-- 【示例 7-5】演示 COALESCE（处理潜在NULL值）
SELECT
    emp_name,
    city,
    COALESCE(city, '未分配') AS city_safe,
    NVL(city, '未知城市')    AS city_nvl
FROM unit7db.employees
LIMIT 5;
```

[SQL]

```
-- 【示例 7-6】NULLIF 演示（防止除零错误）
SELECT
    10 / NULLIF(0, 0) AS safe_divide,   -- 返回 NULL 而非报错
    10 / NULLIF(2, 0) AS normal_divide;
```

7.5 练习题（条件函数）

【练习 7-A】 查询学生成绩表，根据每位学生三科平均分，用 `CASE WHEN` 生成等级（A/B/C/D），并额外用 `IF` 函数判断是否通过（平均分 ≥ 60 则"通过"，否则"补考"），同时显示 `stu_name`、`class_name`、三科成绩、`avg_score`、`grade`、`pass_status`。

[SQL]

```
-- 在此处编写你的SQL语句：
```

【练习 7-B】 查询员工表，根据 `dept_id` 使用 `CASE WHEN` 显示部门名称（10→技术部，20→数据部，30→人力资源部，40→财务部），同时根据在职年数（FLOOR(DATEDIFF/365)）判断：若在职 ≥ 10 年则标注"资深员工"，否则"一般员工"。

[SQL]

```
-- 在此处编写你的SQL语句：
```

8. 聚合函数实践

8.1 目标

- 熟练使用 Hive 聚合函数进行统计分析

- 结合 GROUP BY 进行多维度数据汇总

8.2 时间分配：20 分钟

8.3 函数总览

函数	说明
`COUNT(expr)`	计数
`SUM(expr)`	求和

`AVG(expr)`	平均值
`MAX(expr)`	最大值
`MIN(expr)`	最小值
`STDDEV(expr)`	标准差
`VARIANCE(expr)`	方差
`COLLECT_SET(expr)`	收集不重复值为数组
`COLLECT_LIST(expr)`	收集所有值为数组（含重复）
`PERCENTILE(expr, p)`	百分位数

8.4 实践步骤

步骤 1：基础聚合统计

[SQL]

```
-- 【示例 8-1】员工整体薪资统计
SELECT
    COUNT(*)                    AS total_employees,
    ROUND(AVG(salary), 2)       AS avg_salary,
    MAX(salary)                 AS max_salary,
    MIN(salary)                 AS min_salary,
    SUM(salary)                 AS total_salary_budget,
    ROUND(STDDEV(salary), 2)    AS salary_stddev
FROM unit7db.employees;
```

步骤 2：分组聚合

[SQL]

```
-- 【示例 8-2】按部门统计薪资
SELECT
    dept_id,
    COUNT(*)                    AS headcount,
    ROUND(AVG(salary), 2)       AS avg_salary,
    MAX(salary)                 AS max_salary,
    MIN(salary)                 AS min_salary,
    SUM(salary)                 AS dept_salary_budget
FROM unit7db.employees
GROUP BY dept_id
ORDER BY avg_salary DESC;
```

[SQL]

```
-- 【示例 8-3】按部门和性别统计员工数与平均薪资
SELECT
    dept_id,
    gender,
    COUNT(*)                    AS count,
    ROUND(AVG(salary), 2)       AS avg_salary
FROM unit7db.employees
GROUP BY dept_id, gender
ORDER BY dept_id, gender;
```

步骤 3：COLLECT_SET 与 COLLECT_LIST

[SQL]

```
-- 【示例 8-4】用 COLLECT_SET 收集每个部门的城市列表
SELECT
    dept_id,
    COLLECT_SET(city)           AS cities,
    COLLECT_LIST(emp_name)      AS emp_names
FROM unit7db.employees
GROUP BY dept_id;
```

步骤 4：HAVING 过滤

[SQL]

```
-- 【示例 8-5】筛选平均薪资超过15000的部门
SELECT
    dept_id,
    COUNT(*)                    AS headcount,
    ROUND(AVG(salary), 2)       AS avg_salary
FROM unit7db.employees
GROUP BY dept_id
HAVING AVG(salary) > 15000
ORDER BY avg_salary DESC;
```

[SQL]

```
-- 【示例 8-6】班级成绩统计
SELECT
    class_name,
    COUNT(*)                    AS student_count,
    ROUND(AVG(math_score), 1)   AS avg_math,
```

```
    ROUND(AVG(eng_score), 1)    AS avg_eng,
    ROUND(AVG(sci_score), 1)    AS avg_sci,
    MAX(math_score + eng_score + sci_score) AS max_total,
    MIN(math_score + eng_score + sci_score) AS min_total
FROM unit7db.students
GROUP BY class_name
ORDER BY avg_math DESC;
```

8.5 练习题（聚合函数）

【练习 8-A】 按城市统计订单总金额、订单数量和平均订单金额（保留2位小数），筛选出总金额超过5000元的城市，按总金额降序排列。

[SQL]

```
-- 在此处编写你的SQL语句：
```

【练习 8-B】 使用 `COLLECT_SET` 查询每个班级中出现的所有考试日期（去重），以及每个班级学生姓名列表（用 `COLLECT_LIST`），同时统计各班级的学生人数。

[SQL]

```
-- 在此处编写你的SQL语句：
```

---

9. 窗口函数实践

9.1 目标

- 掌握 Hive 窗口函数（分析函数）的核心用法

- 能实现排名、分组排名、滑动统计等高级分析

9.2 时间分配：35 分钟

9.3 函数总览

函数	说明
`ROW_NUMBER()`	唯一行号（无并列）
`RANK()`	排名（并列跳号）
`DENSE_RANK()`	排名（并列不跳号）
`NTILE(n)`	分桶（平均分为n组）
`LAG(expr, n)`	取前 n 行数据
`LEAD(expr, n)`	取后 n 行数据
`FIRST_VALUE(expr)`	窗口内第一个值
`LAST_VALUE(expr)`	窗口内最后一个值
`SUM/AVG/COUNT OVER`	窗口内聚合
`PERCENT_RANK()`	百分位排名（0~1）
`CUME_DIST()`	累积分布（0~1）

语法格式：

[SQL]

```
函数名() OVER (
    [PARTITION BY 分区列]
    [ORDER BY 排序列]
    [ROWS/RANGE BETWEEN ... AND ...]
)
```

9.4 实践步骤

步骤 1：ROW_NUMBER / RANK / DENSE_RANK

[SQL]

```
-- 【示例 9-1】员工全体薪资排名（三种方式对比）
SELECT
    emp_name,
    salary,
```

```
    ROW_NUMBER() OVER (ORDER BY salary DESC)    AS row_num,
    RANK()       OVER (ORDER BY salary DESC)    AS rank_num,
    DENSE_RANK() OVER (ORDER BY salary DESC)    AS dense_rank_num
FROM unit7db.employees
ORDER BY salary DESC;
```

[SQL]

```
-- 【示例 9-2】按部门内部薪资排名
SELECT
    dept_id,
    emp_name,
    salary,
    RANK()        OVER (PARTITION BY dept_id ORDER BY salary DESC) AS dept_rank,
    DENSE_RANK()  OVER (PARTITION BY dept_id ORDER BY salary DESC) AS dept_dense_rank
FROM unit7db.employees
ORDER BY dept_id, salary DESC;
```

步骤 2：取前 N 名（结合子查询）

[SQL]

```
-- 【示例 9-3】取每个部门薪资前2名的员工
SELECT dept_id, emp_name, salary, dept_rank
FROM (
    SELECT
        dept_id,
        emp_name,
        salary,
        DENSE_RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC) AS dept_rank
    FROM unit7db.employees
) ranked
WHERE dept_rank <= 2
ORDER BY dept_id, dept_rank;
```

步骤 3：NTILE 分桶

[SQL]

```
-- 【示例 9-4】将学生按平均分分为4个等级桶
SELECT
    stu_name,
    class_name,
    ROUND((math_score + eng_score + sci_score) / 3.0, 1) AS avg_score,
    NTILE(4) OVER (ORDER BY (math_score + eng_score + sci_score) DESC) AS score_bucket
```

```
FROM unit7db.students
ORDER BY avg_score DESC;
```

步骤 4：LAG / LEAD 行间比较

[SQL]

```
-- 【示例 9-5】按订单日期查看相邻订单金额变化
SELECT
    order_id,
    order_date,
    total_amount,
    LAG(total_amount, 1, 0) OVER (ORDER BY order_date)  AS prev_amount,
    LEAD(total_amount, 1, 0) OVER (ORDER BY order_date) AS next_amount,
    ROUND(total_amount - LAG(total_amount, 1, 0) OVER (ORDER BY order_date), 2) AS
amount_change
FROM unit7db.orders
ORDER BY order_date;
```

步骤 5：窗口聚合（累计/滑动）

[SQL]

```
-- 【示例 9-6】订单金额累计求和（按日期排序）
SELECT
    order_id,
    order_date,
    total_amount,
    SUM(total_amount) OVER (ORDER BY order_date
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)    AS cumulative_sum,
    AVG(total_amount) OVER (ORDER BY order_date
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)           AS moving_avg_3
FROM unit7db.orders
ORDER BY order_date;
```

[SQL]

```
-- 【示例 9-7】每位员工薪资与部门平均薪资的差值
SELECT
    dept_id,
    emp_name,
    salary,
    ROUND(AVG(salary) OVER (PARTITION BY dept_id), 2) AS dept_avg_salary,
    ROUND(salary - AVG(salary) OVER (PARTITION BY dept_id), 2) AS diff_from_dept_avg,
    SUM(salary)   OVER (PARTITION BY dept_id) AS dept_total_salary
FROM unit7db.employees
```

```
ORDER BY dept_id, salary DESC;
```

步骤 6：FIRST_VALUE / LAST_VALUE

[SQL]

```
-- 【示例 9-8】每个部门中薪资最高员工的薪资（窗口函数方式）
SELECT
    dept_id,
    emp_name,
    salary,
    FIRST_VALUE(emp_name) OVER (PARTITION BY dept_id ORDER BY salary DESC
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS top_earner_in_dept,
    FIRST_VALUE(salary) OVER (PARTITION BY dept_id ORDER BY salary DESC
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS top_salary_in_dept
FROM unit7db.employees
ORDER BY dept_id, salary DESC;
```

9.5 练习题（窗口函数）

【练习 9-A】 对学生表按班级内部进行三科总分排名（使用 `DENSE_RANK`），显示 `stu_name`、`class_name`、`total_score`（三科之和）、`class_rank`（班内排名）。筛选出每班排名前2名的学生。

[SQL]

```
-- 在此处编写你的SQL语句：
```

【练习 9-B】 对订单表按 `order_date` 排序，使用 `LAG` 函数计算每笔订单与前一笔订单的金额差（`amount_change`），并用 `IF` 判断金额是否增加（增加显示"↑上升"，减少显示"↓下降"，相等显示"→持平"）。

[SQL]

```
-- 在此处编写你的SQL语句：
```

【练习 9-C】 使用窗口函数，计算每位员工的薪资在**全体员工**中的百分位排名（`PERCENT_RANK()`），结果保留4位小数，并按百分位降序显示。

[SQL]

```
-- 在此处编写你的SQL语句：
```

10. 复杂类型函数实践

10.1 目标

- 理解并运用 Hive 的 ARRAY、MAP 类型

- 掌握 EXPLODE、LATERAL VIEW 等高级用法

10.2 时间分配：20 分钟

10.3 函数总览

函数	说明
`ARRAY(v1, v2, ...)`	创建数组
`ARRAY_CONTAINS(arr, val)`	数组是否包含值
`SIZE(arr/map)`	返回数组/MAP大小
`SORT_ARRAY(arr)`	数组排序
`MAP(k1, v1, k2, v2, ...)`	创建MAP
`MAP_KEYS(map)`	返回所有KEY
`MAP_VALUES(map)`	返回所有VALUE
`EXPLODE(arr/map)`	行列转换（展开数组/MAP）
`LATERAL VIEW EXPLODE(...)`	配合原表列一起展开
`POSEXPLODE(arr)`	带下标的 EXPLODE

`STR_TO_MAP(str, del1, del2)`	字符串转 MAP

10.4 实践步骤

步骤 1：ARRAY 操作

[SQL]

```
-- 【示例 10-1】创建和操作数组
SELECT
    stu_name,
    ARRAY(math_score, eng_score, sci_score)                    AS scores_array,
    SIZE(ARRAY(math_score, eng_score, sci_score))              AS array_size,
    ARRAY_CONTAINS(ARRAY(math_score, eng_score, sci_score), 90) AS has_score_90,
    SORT_ARRAY(ARRAY(math_score, eng_score, sci_score))        AS sorted_scores
FROM unit7db.students
LIMIT 5;
```

步骤 2：EXPLODE 展开

[SQL]

```
-- 【示例 10-2】用 COLLECT_SET + EXPLODE 演示行列互转
-- 先收集每部门城市
SELECT dept_id, COLLECT_SET(city) AS cities
FROM unit7db.employees
GROUP BY dept_id;
```

[SQL]

```
-- 【示例 10-3】EXPLODE 展开内联数组
SELECT exploded_score
FROM (SELECT ARRAY(92, 85, 88) AS scores) t
LATERAL VIEW EXPLODE(scores) score_table AS exploded_score;
```

[SQL]

```
-- 【示例 10-4】LATERAL VIEW EXPLODE 与原表联合
SELECT
    stu_name,
    score_type,
    score_value
```

```
FROM unit7db.students
LATERAL VIEW EXPLODE(
    MAP('数学', math_score, '英语', eng_score, '科学', sci_score)
) score_table AS score_type, score_value
ORDER BY stu_name, score_type;
```

步骤 3：MAP 操作

[SQL]

```
-- 【示例 10-5】MAP 创建与操作
SELECT
    emp_name,
    MAP('薪资', salary, '部门', dept_id)                AS emp_map,
    MAP_KEYS(MAP('薪资', salary, '部门', dept_id))       AS map_keys,
    MAP_VALUES(MAP('薪资', salary, '部门', dept_id))     AS map_values
FROM unit7db.employees
LIMIT 3;
```

步骤 4：STR_TO_MAP

[SQL]

```
-- 【示例 10-6】字符串转MAP
SELECT
    STR_TO_MAP('name=张三,age=25,city=北京', ',', '=') AS str_map,
    STR_TO_MAP('name=张三,age=25,city=北京', ',', '=')['name'] AS extracted_name;
```

10.5 练习题（复杂类型函数）

【练习 10-A】 使用 `LATERAL VIEW EXPLODE` 将学生表中每位学生的三科成绩（以MAP形式：'数学', math_score, '英语', eng_score, '科学', sci_score）展开成多行，每行包含 `stu_name`、`class_name`、`subject`（科目）、`score`（分数），并筛选出分数低于 70 分的记录。

[SQL]

```
-- 在此处编写你的SQL语句：
```

11. 类型转换函数实践

11.1 目标

- 掌握 Hive 数据类型转换

- 处理常见的类型不匹配问题

11.2 时间分配：15 分钟

11.3 函数总览

函数	说明
`CAST(expr AS type)`	显式类型转换
`BINARY(str)`	转为二进制
`TO_DATE(str)`	字符串转日期
`DATE_FORMAT(date, fmt)`	日期转字符串

常用类型名称： `INT`、`BIGINT`、`DOUBLE`、`FLOAT`、`STRING`、`BOOLEAN`、`DATE`、`TIMESTAMP`

11.4 实践步骤

[SQL]

```
-- 【示例 11-1】CAST 基础演示
SELECT
    CAST('123' AS INT)          AS str_to_int,
    CAST(456 AS STRING)         AS int_to_str,
    CAST('3.14' AS DOUBLE)      AS str_to_double,
    CAST(1 AS BOOLEAN)          AS int_to_bool,
    CAST('2024-01-01' AS DATE)  AS str_to_date;
```

[SQL]

```
-- 【示例 11-2】实际场景：薪资字符串处理
SELECT
    emp_name,
```

```
    salary,
    CAST(salary AS INT)                                 AS salary_int,
    CAST(salary AS STRING)                              AS salary_str,
    LENGTH(CAST(salary AS STRING))                      AS salary_digits,
    CONCAT('¥', CAST(ROUND(salary/1000, 1) AS STRING), 'K') AS salary_display
FROM unit7db.employees
LIMIT 8;
```

[SQL]

```
-- 【示例 11-3】日期类型转换
SELECT
    emp_name,
    hire_date,
    CAST(hire_date AS DATE)                             AS hire_date_typed,
    CAST(hire_date AS TIMESTAMP)                        AS hire_timestamp,
    DATE_FORMAT(CAST(hire_date AS DATE), 'MM/dd/yyyy') AS hire_date_us_format,
    UNIX_TIMESTAMP(hire_date, 'yyyy-MM-dd')             AS hire_unix_ts
FROM unit7db.employees
LIMIT 5;
```

[SQL]

```
-- 【示例 11-4】类型转换注意事项
SELECT
    CAST('abc' AS INT)          AS invalid_cast,       -- 返回 NULL
    CAST(NULL AS STRING)        AS null_to_str,        -- 返回 NULL
    CAST(TRUE AS INT)           AS bool_to_int,        -- 返回 1
    CAST(FALSE AS INT)          AS false_to_int;       -- 返回 0
```

11.5 练习题（类型转换）

【练习 11-A】 查询员工表，将薪资格式化为 "¥XX,XXX.XX 元" 的字符串显示格式（使用 `CAST`、`FORMAT_NUMBER` 或 `CONCAT` 组合实现），同时显示员工编号（6位零填充格式）、姓名和格式化薪资。

[SQL]

```
-- 在此处编写你的SQL语句：
```

12. 综合练习题

12.1 目标

- 综合运用本章所学的多种函数

- 解决贴近实际业务的复杂分析问题

12.2 时间分配：35 分钟

【综合练习 1】员工综合信息报告

难度：★★☆☆☆ 预计 7 分钟

查询员工表，生成如下格式的员工信息报告。要求：

1. 生成员工编号（6位零填充）

2. 显示员工姓名及性别（M→男，F→女）

3. 计算年龄（基于 birth_date）

4. 计算在职年数

5. 显示薪资等级（参考示例 7-2）

6. 按薪资降序排列

输出列：`formatted_id`、`emp_name`、`gender_cn`、`age`、`years_worked`、`salary`、`salary_grade`

[SQL]

```
-- 在此处编写你的SQL语句：
```

【综合练习 2】订单城市销售排名分析

难度：★★★☆☆ 预计 8 分钟

统计每个城市的订单总金额，计算每个城市的总金额排名（`RANK`），并计算该城市总金额占全部订单总金额的百分比（保留2位小数，格式如 "23.45%"）。筛选出排名前3的城市。

输出列：`city`、`total_amount`、`city_rank`、`pct_of_total`

[SQL]

```sql
-- 在此处编写你的SQL语句：
```

【综合练习 3】部门薪资深度分析

难度：★★★★☆ 预计 10 分钟

对员工表进行部门薪资深度分析，要求：

1. 关联 departments 表显示部门名称

2. 显示每位员工的薪资、部门平均薪资

3. 计算员工薪资与部门平均薪资的差值，并注明高于/低于平均薪资

4. 显示员工在部门内的薪资排名（DENSE_RANK）

5. 显示部门内薪资最高员工的姓名（FIRST_VALUE）

输出列：`dept_name`、`emp_name`、`salary`、`dept_avg`、`diff`、`above_below`、`dept_rank`、`dept_top_earner`

[SQL]

```sql
-- 在此处编写你的SQL语句：
```

【综合练习 4】班级成绩全面分析报告

难度：★★★★☆ 预计 10 分钟

对学生表进行综合分析：

1. 计算每位学生的总分和平均分

2. 按总分给出等级（A/B/C/D）

3. 计算每位学生在全体学生中的总分排名（DENSE_RANK）

4. 计算每位学生总分与全体平均总分的差值

5. 用 `COLLECT_LIST` 显示每个班级中达到A级的学生名单（子查询）

[SQL]

```sql
-- 步骤一：学生个人分析


-- 步骤二：班级A级学生名单
```

---

13. 课堂总结与评分标准

13.1 本章函数总结表

函数类别	重要函数	应用场景
字符串函数	CONCAT_WS, SUBSTR, INSTR, REGEXP_REPLACE	数据清洗、字段格式化
数值函数	ROUND, FLOOR, CEIL, MOD, GREATEST	财务计算、数据分级
日期函数	DATEDIFF, DATE_ADD, DATE_FORMAT, YEAR/MONTH/DAY	时间分析、报表生成
条件函数	IF, CASE WHEN, COALESCE, NVL	数据分类、标签化
聚合函数	COUNT, SUM, AVG, COLLECT_SET, PERCENTILE	汇总统计、分组分析
窗口函数	ROW_NUMBER, RANK, DENSE_RANK, LAG, LEAD, SUM OVER	排名分析、趋势分析
复杂类型	ARRAY, MAP, EXPLODE, LATERAL VIEW	多值处理、行列转换
类型转换	CAST	类型统一、格式化

13.2 评分标准

评分项目	分值	说明
练习 4-A/B/C（字符串）	9分	各3分

练习 5-A/B（数值）	6分	各3分
练习 6-A/B（日期）	6分	各3分
练习 7-A/B（条件）	6分	各3分
练习 8-A/B（聚合）	6分	各3分
练习 9-A/B/C（窗口）	9分	各3分
练习 10-A（复杂类型）	4分	全对4分
练习 11-A（类型转换）	4分	全对4分
综合练习 1~4	20分	各5分
提交规范性	10分	格式整洁、注释清晰
合计	**80分**	基础分

> 💡 提示： 完成所有综合练习并加以注释说明可获得额外加分。

13.3 提交要求

1. 将本次实践中所有 SQL 语句（含练习题）整理到一个文本文件，命名格式为：`学号_姓名_unit7.sql`
2. 在每道练习题前保留题目注释
3. 确保所有 SQL 语句可以在 `unit7db` 数据库下正常执行
