Apache Hive 第 7 章 函数综合实践 — 教师答案手册

> 注意：本文件为教师专用答案手册，请勿发放给学生。 - 数据库：`unit7db` - 数据集目录：`/usr/datas/unit7/` - Beeline 命令：`hive-bee`

目录

- [第4章 字符串函数 练习答案](#第4章-字符串函数-练习答案)
- [第5章 数值函数 练习答案](#第5章-数值函数-练习答案)
- [第6章 日期函数 练习答案](#第6章-日期函数-练习答案)
- [第7章 条件函数 练习答案](#第7章-条件函数-练习答案)
- [第8章 聚合函数 练习答案](#第8章-聚合函数-练习答案)
- [第9章 窗口函数 练习答案](#第9章-窗口函数-练习答案)
- [第10章 复杂类型函数 练习答案](#第10章-复杂类型函数-练习答案)
- [第11章 类型转换函数 练习答案](#第11章-类型转换函数-练习答案)
- [综合练习 答案](#综合练习-答案)
- [评分注意事项](#评分注意事项)

第 4 章 字符串函数 练习答案

【练习 4-A】答案

题目要求： 查询所有员工邮箱地址，提取 @ 之前的用户名，转为全大写。

[SQL]

```
SELECT
    emp_name,
    email,
    UPPER(SUBSTR(email, 1, INSTR(email, '@') - 1)) AS email_username
FROM unit7db.employees;
```

预期结果（部分）：

emp_name	email	email_username
张伟	zhangwei@company.com	ZHANGWEI
李娜	lina@company.com	LINA
王强	wangqiang@company.com	WANGQIANG

评分要点：

- `INSTR(email, '@') - 1` 计算 @ 前的长度 ✓

- `SUBSTR(email, 1, ...)` 从第1位截取 ✓

- `UPPER(...)` 转大写 ✓

- 列名使用 `email_username` ✓

常见错误：

- `SUBSTR(email, 1, INSTR(email, '@'))` — 多截取了 @ 符号本身（扣0.5分）

- 忘记 `UPPER()` 函数（扣1分）

【练习 4-B】答案

题目要求： 用 " - " 分隔符拼接 `emp_name`、`job_title`、`city`，并计算 `profile` 字段总字符长度。

[SQL]

```
SELECT
    emp_name,
    job_title,
    city,
    CONCAT_WS(' - ', emp_name, job_title, city)         AS profile,
    LENGTH(CONCAT_WS(' - ', emp_name, job_title, city)) AS profile_length
FROM unit7db.employees;
```

预期结果（部分）：

emp_name	job_title	city	profile	profile_length
张伟	软件工程师	北京	张伟 - 软件工程师 - 北京	14
李娜	数据分析师	上海	李娜 - 数据分析师 - 上海	14

> 注：中文字符在 LENGTH() 中按 UTF-8 字节计算（1个汉字=3字节），具体结果取决于 Hive 配置，接受字节数或字符数均可。

评分要点：

- 使用 `CONCAT_WS(' - ', ...)` 分隔符正确 ✓
- `LENGTH()` 函数包裹 profile 字段 ✓
- 输出 profile 和 profile_length 两列 ✓

【练习 4-C】答案

题目要求： 将产品名称中的"A"替换为"Pro"，将"B"替换为"Max"。

```sql
SELECT
    product_name                                              AS original_name,
    REPLACE(REPLACE(product_name, 'A', 'Pro'), 'B', 'Max') AS new_name
FROM unit7db.products;
```

预期结果：

original_name	new_name
笔记本电脑A	笔记本电脑Pro
无线鼠标	无线鼠标
机械键盘	机械键盘
4K显示器	4K显示器
游戏本B	游戏本Max

评分要点：

- 嵌套 `REPLACE()` 进行两次替换 ✓
- 替换值正确（A→Pro, B→Max） ✓

可接受的替代写法：

```sql
-- 使用 REGEXP_REPLACE（同样正确）
SELECT
```

```
    product_name,
    REGEXP_REPLACE(REGEXP_REPLACE(product_name, 'A$', 'Pro'), 'B$', 'Max') AS new_name
FROM unit7db.products;
```

第 5 章 数值函数 练习答案

【练习 5-A】答案

题目要求： 计算订单明细含折扣后实付金额，保留2位小数，按降序排列，显示前8条。

[SQL]

```
SELECT
    item_id,
    order_id,
    product_id,
    quantity,
    unit_price,
    discount,
    ROUND(quantity * unit_price * (1 - discount), 2) AS actual_amount
FROM unit7db.order_items
ORDER BY actual_amount DESC
LIMIT 8;
```

预期结果（部分）：

item_id	order_id	product_id	quantity	unit_price	discount	actual_amount
7	2005	105	2	5500.00	0.05	10450.00
14	2011	105	1	5500.00	0.05	5225.00
12	2009	105	1	5500.00	0.02	5390.00

评分要点：

- 计算公式正确：`quantity * unit_price * (1 - discount)` ✓
- `ROUND(..., 2)` 保留2位小数 ✓
- `ORDER BY actual_amount DESC` 降序 ✓
- `LIMIT 8` ✓

【练习 5-B】答案

题目要求： 对 `salary / 1000` 分别用 `ROUND`、`FLOOR`、`CEIL` 取整。

[SQL]

```
SELECT
    emp_name,
    salary,
    ROUND(salary / 1000, 1)     AS salary_k_round,
    FLOOR(salary / 1000)        AS salary_k_floor,
    CEIL(salary / 1000)         AS salary_k_ceil
FROM unit7db.employees
ORDER BY salary DESC;
```

预期结果（部分）：

emp_name	salary	salary_k_round	salary_k_floor	salary_k_ceil
黄磊	40000.0	40.0	40	40
胡军	35000.0	35.0	35	35
陈明	25000.0	25.0	25	25
王强	20000.0	20.0	20	20
张伟	15000.0	15.0	15	15
吴刚	14000.0	14.0	14	14

评分要点：

- 三种取整函数均使用 ✓

- `ROUND` 参数指定小数位 ✓

- 列名合理命名 ✓

第 6 章 日期函数 练习答案

【练习 6-A】答案

题目要求： 计算员工在职年数，以及下一个入职周年纪念日。

[SQL]

```
SELECT
    emp_name,
    hire_date,
    FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365)  AS years_worked,
    DATE_ADD(
        hire_date,
        (FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365) + 1) * 365
    )                                                AS next_anniversary
FROM unit7db.employees
ORDER BY years_worked DESC;
```

预期结果（部分，基于2024年计算）：

emp_name	hire_date	years_worked	next_anniversary
黄磊	2007-10-01	16	...
胡军	2009-01-01	15	...
张伟	2010-06-01	14	...

> 注：由于 `CURRENT_DATE` 是动态的，具体数值会随执行日期变化，只需逻辑正确即可。

评分要点：

- `FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365)` 计算在职年数 ✓
- `DATE_ADD(hire_date, (years+1)*365)` 计算周年纪念日 ✓（可使用子查询或嵌套）
- 按 `years_worked` 降序排列 ✓

可接受的替代写法（使用子查询）：

[SQL]

```
SELECT
    emp_name,
    hire_date,
    years_worked,
    DATE_ADD(hire_date, (years_worked + 1) * 365) AS next_anniversary
FROM (
    SELECT
        emp_name,
        hire_date,
        FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365) AS years_worked
    FROM unit7db.employees
) t
ORDER BY years_worked DESC;
```

【练习 6-B】答案

题目要求： 统计 2024年每月订单总金额和订单数量，按月份排序。

[SQL]

```
SELECT
    DATE_FORMAT(order_date, 'yyyy-MM')  AS order_month,
    COUNT(*)                            AS order_count,
    ROUND(SUM(total_amount), 2)         AS total_revenue
FROM unit7db.orders
WHERE YEAR(order_date) = 2024
GROUP BY DATE_FORMAT(order_date, 'yyyy-MM')
ORDER BY order_month;
```

预期结果：

order_month	order_count	total_revenue
2024-01	5	20300.49
2024-02	5	11130.50
2024-03	5	21530.00

评分要点：

- `DATE_FORMAT(order_date, 'yyyy-MM')` 格式正确 ✓
- `WHERE YEAR(order_date) = 2024` 或等价过滤 ✓
- `ROUND(SUM(...), 2)` 保留2位小数 ✓
- `GROUP BY` 与 `SELECT` 中的月份表达式一致 ✓
- `ORDER BY order_month` ✓

第 7 章 条件函数 练习答案

【练习 7-A】答案

题目要求： 生成学生等级（A/B/C/D）和通过状态。

[SQL]

```
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'
        WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 75 THEN 'B'
        WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 60 THEN 'C'
        ELSE 'D'
    END AS grade,
    IF(
        ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 60,
        '通过',
        '补考'
    ) AS pass_status
FROM unit7db.students
ORDER BY avg_score DESC;
```

预期结果：

stu_name	class_name	math_score	eng_score	sci_score	avg_score	grade	pass_status
秦小强	大数据2201	98	96	99	97.7	A	通过
赵小雅	大数据2202	95	88	92	91.7	A	通过
郑雅丽	大数据2202	90	95	91	92.0	A	通过
楚天舒	大数据2202	45	55	50	50.0	D	补考
孙晓光	大数据2201	55	60	58	57.7	D	补考

评分要点：

- `CASE WHEN` 四个等级边界正确（90/75/60）✓
- `IF(avg >= 60, '通过', '补考')` ✓
- 输出所有要求列 ✓

【练习 7-B】答案

题目要求： 显示部门名称，并判断员工类型（资深/一般）。

[SQL]

```
SELECT
    emp_name,
    CASE dept_id
        WHEN 10 THEN '技术部'
        WHEN 20 THEN '数据部'
        WHEN 30 THEN '人力资源部'
        WHEN 40 THEN '财务部'
        ELSE '未知部门'
    END AS dept_name,
    salary,
    FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365) AS years_worked,
    IF(
        FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365) >= 10,
        '资深员工',
        '一般员工'
    ) AS employee_type
FROM unit7db.employees
ORDER BY years_worked DESC;
```

预期结果（部分）：

emp_name	dept_name	salary	years_worked	employee_type
黄磊	技术部	40000.0	16+	资深员工
胡军	财务部	35000.0	15+	资深员工
刘芳	技术部	8000.0	4	一般员工

评分要点：

- `CASE dept_id WHEN ... THEN ...` 四个部门均映射 ✓
- `FLOOR(DATEDIFF/365)` 计算在职年数 ✓
- `IF(years >= 10, '资深员工', '一般员工')` ✓

第 8 章 聚合函数 练习答案

【练习 8-A】答案

题目要求： 按城市统计订单总金额、订单数，筛选总金额 > 5000，降序排列。

```sql
SELECT
    city,
    COUNT(*)                    AS order_count,
    ROUND(SUM(total_amount), 2) AS total_amount,
    ROUND(AVG(total_amount), 2) AS avg_amount
FROM unit7db.orders
GROUP BY city
HAVING SUM(total_amount) > 5000
ORDER BY total_amount DESC;
```

预期结果：

city	order_count	total_amount	avg_amount
北京	6	37580.49	6263.42
深圳	3	15380.00	5126.67
广州	3	8050.00	2683.33
上海	3	5070.50	1690.17

> 注：具体数值以实际加载数据为准。

评分要点：

- `GROUP BY city` ✓
- `HAVING SUM(total_amount) > 5000` ✓（注意不能用 WHERE 替代 HAVING）
- `ORDER BY total_amount DESC` ✓
- 三个聚合函数均使用 ✓

【练习 8-B】答案

题目要求： 使用 `COLLECT_SET` 和 `COLLECT_LIST` 查询班级考试日期和学生名单。

```sql
SELECT
    class_name,
    COUNT(*)                    AS student_count,
```

```
    COLLECT_SET(exam_date)      AS exam_dates,
    COLLECT_LIST(stu_name)      AS student_names
FROM unit7db.students
GROUP BY class_name
ORDER BY class_name;
```

预期结果：

class_name	student_count	exam_dates	student_names
大数据2201	4	["2024-01-15"]	["张小明","李小红","孙晓光","冯建国","秦小强"]
大数据2202	4	["2024-01-15"]	["王大壮","赵小雅","郑雅丽","楚天舒"]
大数据2203	3	["2024-01-15"]	["周佳佳","吴小波","陈晓丹"]

> 注：`COLLECT_SET` 去除重复值，`COLLECT_LIST` 保留所有值。

评分要点：

- `COLLECT_SET(exam_date)` 与 `COLLECT_LIST(stu_name)` 正确区分 ✓
- `COUNT(*)` 统计学生数 ✓
- `GROUP BY class_name` ✓

第 9 章 窗口函数 练习答案

【练习 9-A】答案

题目要求： 按班级内总分排名，取每班前2名。

[SQL]

```
SELECT
    stu_name,
    class_name,
    total_score,
    class_rank
FROM (
    SELECT
```

```
        stu_name,
        class_name,
        math_score + eng_score + sci_score                          AS total_score,
        DENSE_RANK() OVER (
            PARTITION BY class_name
            ORDER BY (math_score + eng_score + sci_score) DESC
        )                                                           AS class_rank
    FROM unit7db.students
) ranked
WHERE class_rank <= 2
ORDER BY class_name, class_rank;
```

预期结果：

stu_name	class_name	total_score	class_rank
秦小强	大数据2201	293	1
冯建国	大数据2201	241	2
郑雅丽	大数据2202	276	1
赵小雅	大数据2202	275	2
周佳佳	大数据2203	265	1
吴小波	大数据2203	218	2

评分要点：

- `PARTITION BY class_name` 按班级分区 ✓
- `ORDER BY total_score DESC` 降序 ✓
- 使用子查询过滤 `WHERE class_rank <= 2` ✓
- 使用 `DENSE_RANK()` 而非 `RANK()` ✓（RANK也接受，但DENSE_RANK更合适）

【练习 9-B】答案

题目要求： LAG 计算订单金额变化，IF 判断方向。

[SQL]

```
SELECT
    order_id,
    order_date,
    total_amount,
```

```
    LAG(total_amount, 1, 0) OVER (ORDER BY order_date) AS prev_amount,
    ROUND(total_amount - LAG(total_amount, 1, 0) OVER (ORDER BY order_date), 2) AS
amount_change,
    IF(
        total_amount > LAG(total_amount, 1, NULL) OVER (ORDER BY order_date),
        '↑上升',
        IF(
            total_amount < LAG(total_amount, 1, NULL) OVER (ORDER BY order_date),
            '↓下降',
            '→持平'
        )
    ) AS trend
FROM unit7db.orders
ORDER BY order_date;
```

评分要点：

- `LAG(total_amount, 1, ...)` 正确取前一行 ✓

- 计算 `amount_change` = 当前 - 前一行 ✓

- 三种趋势判断（上升/下降/持平）均覆盖 ✓

- 第一行（无前值）：`LAG` 返回默认值 NULL 或 0，趋势显示为"↓下降"或"→持平"均可接受

简化接受版本：

[SQL]

```
SELECT
    order_id,
    order_date,
    total_amount,
    ROUND(total_amount - LAG(total_amount, 1, total_amount) OVER (ORDER BY order_date), 2)
AS amount_change,
    CASE
        WHEN total_amount > LAG(total_amount, 1, total_amount) OVER (ORDER BY order_date)
THEN '↑上升'
        WHEN total_amount < LAG(total_amount, 1, total_amount) OVER (ORDER BY order_date)
THEN '↓下降'
        ELSE '→持平'
    END AS trend
FROM unit7db.orders
ORDER BY order_date;
```

【练习 9-C】答案

题目要求： 计算员工薪资的百分位排名。

```sql
SELECT
    emp_name,
    salary,
    ROUND(
        PERCENT_RANK() OVER (ORDER BY salary),
        4
    ) AS percent_rank_val
FROM unit7db.employees
ORDER BY percent_rank_val DESC;
```

预期结果（部分）：

emp_name	salary	percent_rank_val
黄磊	40000.0	1.0000
胡军	35000.0	0.9286
陈明	25000.0	0.8571
刘芳	8000.0	0.0714
林晓	6000.0	0.0000

计算原理： `PERCENT_RANK = (rank - 1) / (total_rows - 1)`，范围 [0, 1]

评分要点：

- `PERCENT_RANK()` 函数使用正确 ✓
- `OVER (ORDER BY salary)` ✓
- `ROUND(..., 4)` 保留4位小数 ✓
- `ORDER BY percent_rank_val DESC` ✓

第 10 章 复杂类型函数 练习答案

【练习 10-A】答案

题目要求： 用 LATERAL VIEW EXPLODE 展开学生三科成绩，筛选低于70分的记录。

[SQL]

```
SELECT
    stu_name,
    class_name,
    subject,
    score
FROM unit7db.students
LATERAL VIEW EXPLODE(
    MAP('数学', math_score, '英语', eng_score, '科学', sci_score)
) score_table AS subject, score
WHERE score < 70
ORDER BY stu_name, subject;
```

预期结果（部分）：

stu_name	class_name	subject	score
王大壮	大数据2202	数学	65
孙晓光	大数据2201	数学	55
孙晓光	大数据2201	英语	60
孙晓光	大数据2201	科学	58
楚天舒	大数据2202	数学	45
楚天舒	大数据2202	英语	55
楚天舒	大数据2202	科学	50

评分要点：

- `LATERAL VIEW EXPLODE(MAP(...))` 语法正确 ✓
- MAP中三科均包含 ✓
- `AS subject, score` 别名正确 ✓
- `WHERE score < 70` 过滤条件 ✓

第 11 章 类型转换函数 练习答案

【练习 11-A】答案

题目要求： 员工编号6位零填充，薪资格式化为可读格式。

[SQL]

```
SELECT
    LPAD(CAST(emp_id AS STRING), 6, '0')    AS formatted_emp_id,
    emp_name,
    CONCAT('¥', FORMAT_NUMBER(salary, 2), ' 元') AS formatted_salary
FROM unit7db.employees
ORDER BY salary DESC;
```

> 如果 `FORMAT_NUMBER` 不可用，使用以下替代方案：

[SQL]

```
-- 替代方案（不使用 FORMAT_NUMBER）
SELECT
    LPAD(CAST(emp_id AS STRING), 6, '0')    AS formatted_emp_id,
    emp_name,
    salary,
    CONCAT('¥', CAST(CAST(salary AS BIGINT) AS STRING), '.00 元') AS formatted_salary
FROM unit7db.employees
ORDER BY salary DESC;
```

预期结果（部分）：

formatted_emp_id	emp_name	formatted_salary
001015	黄磊	¥40,000.00 元
001013	胡军	¥35,000.00 元

评分要点：

- `LPAD(CAST(emp_id AS STRING), 6, '0')` 正确填充6位 ✓
- `CAST` 类型转换使用正确 ✓
- 薪资格式化包含货币符号和单位 ✓

综合练习 答案

【综合练习 1】员工综合信息报告 答案

[SQL]

```
SELECT
    LPAD(CAST(emp_id AS STRING), 6, '0')                AS formatted_id,
    emp_name,
    IF(gender = 'M', '男', '女')                          AS gender_cn,
    FLOOR(DATEDIFF(CURRENT_DATE, birth_date) / 365)     AS age,
    FLOOR(DATEDIFF(CURRENT_DATE, hire_date) / 365)      AS years_worked,
    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 '基础/实习'
    END AS salary_grade
FROM unit7db.employees
ORDER BY salary DESC;
```

预期结果（部分）：

formatted_id	emp_name	gender_cn	age	years_worked	salary	salary_grade
001015	黄磊	男	42	16	40000.0	高管级
001013	胡军	男	40	15	35000.0	高管级
001005	陈明	男	38	13	25000.0	高级

评分要点（5分）：

- `LPAD` 格式化编号 (1分)
- `IF(gender, ...)` 性别转换 (1分)
- `FLOOR(DATEDIFF/365)` 年龄/在职年数 (1分)
- `CASE WHEN` 薪资等级5级正确 (1分)
- `ORDER BY salary DESC` (1分)

【综合练习 2】订单城市销售排名分析 答案

[SQL]

```
SELECT
```

```
    city,
    ROUND(SUM(total_amount), 2)                          AS total_amount,
    RANK() OVER (ORDER BY SUM(total_amount) DESC)        AS city_rank,
    CONCAT(
        CAST(ROUND(SUM(total_amount) /
            SUM(SUM(total_amount)) OVER () * 100, 2) AS STRING),
        '%'
    )                                                    AS pct_of_total
FROM unit7db.orders
GROUP BY city
QUALIFY RANK() OVER (ORDER BY SUM(total_amount) DESC) <= 3
ORDER BY city_rank;
```

> 注意：Hive 不支持 `QUALIFY` 子句，需使用子查询：

[SQL]

```
-- 标准 Hive 写法
SELECT city, total_amount, city_rank, pct_of_total
FROM (
    SELECT
        city,
        ROUND(SUM(total_amount), 2) AS total_amount,
        RANK() OVER (ORDER BY SUM(total_amount) DESC) AS city_rank,
        CONCAT(
            CAST(ROUND(
                SUM(total_amount) / SUM(SUM(total_amount)) OVER () * 100, 2
            ) AS STRING),
            '%'
        ) AS pct_of_total
    FROM unit7db.orders
    GROUP BY city
) ranked
WHERE city_rank <= 3
ORDER BY city_rank;
```

预期结果：

city	total_amount	city_rank	pct_of_total
北京	37580.49	1	~67.xx%
深圳	15380.00	2	~27.xx%
广州	8050.00	3	~14.xx%

评分要点（5分）：

- `GROUP BY city` + `SUM` 聚合 (1分)

- `RANK() OVER (ORDER BY SUM(...) DESC)` (1分)

- `SUM(SUM(...)) OVER ()` 嵌套窗口函数计算总计 (1分)

- 百分比格式化（含%符号）(1分)

- 子查询过滤前3名 (1分)

【综合练习 3】部门薪资深度分析 答案

[SQL]

```
SELECT
    d.dept_name,
    e.emp_name,
    e.salary,
    ROUND(AVG(e.salary) OVER (PARTITION BY e.dept_id), 2)                    AS
dept_avg,
    ROUND(e.salary - AVG(e.salary) OVER (PARTITION BY e.dept_id), 2)         AS diff,
    IF(
        e.salary >= AVG(e.salary) OVER (PARTITION BY e.dept_id),
        '高于平均',
        '低于平均'
    )                                                                         AS
above_below,
    DENSE_RANK() OVER (PARTITION BY e.dept_id ORDER BY e.salary DESC)        AS
dept_rank,
    FIRST_VALUE(e.emp_name) OVER (
        PARTITION BY e.dept_id
        ORDER BY e.salary DESC
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
    )                                                                         AS
dept_top_earner
FROM unit7db.employees e
JOIN unit7db.departments d ON e.dept_id = d.dept_id
ORDER BY d.dept_name, e.salary DESC;
```

预期结果（部分）：

dept_name	emp_name	salary	dept_avg	diff	above_below	dept_rank	dept_top_earner
技术部	黄磊	40000	20000	20000	高于平均	1	黄磊
技术部	王强	20000	20000	0	高于平均	2	黄磊
技术部	张伟	15000	20000	-5000	低于平均	3	黄磊

评分要点（5分）：

- `JOIN departments` 关联正确 (1分)
- `AVG() OVER (PARTITION BY dept_id)` (1分)
- `IF(salary >= dept_avg, '高于', '低于')` (1分)
- `DENSE_RANK() OVER PARTITION BY dept_id` (1分)
- `FIRST_VALUE() OVER ... ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING` (1分)

【综合练习 4】班级成绩全面分析报告 答案

步骤一：学生个人分析

[SQL]

```
SELECT
    stu_name,
    class_name,
    math_score + eng_score + sci_score      AS total_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'
        WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 75 THEN 'B'
        WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 60 THEN 'C'
        ELSE 'D'
    END AS grade,
    DENSE_RANK() OVER (ORDER BY (math_score + eng_score + sci_score) DESC) AS
overall_rank,
    ROUND(
        (math_score + eng_score + sci_score) -
        AVG(math_score + eng_score + sci_score) OVER (),
        1
    )                                       AS diff_from_avg
FROM unit7db.students
ORDER BY total_score DESC;
```

步骤二：班级A级学生名单

[SQL]

```
SELECT
    class_name,
    COLLECT_LIST(stu_name) AS grade_a_students,
    COUNT(*)               AS grade_a_count
```

```
FROM (
    SELECT
        stu_name,
        class_name,
        CASE
            WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 90 THEN 'A'
            WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 75 THEN 'B'
            WHEN ROUND((math_score + eng_score + sci_score) / 3.0, 1) >= 60 THEN 'C'
            ELSE 'D'
        END AS grade
    FROM unit7db.students
) graded
WHERE grade = 'A'
GROUP BY class_name
ORDER BY class_name;
```

预期结果（步骤二）：

class_name	grade_a_students	grade_a_count
大数据2201	["秦小强"]	1
大数据2202	["赵小雅","郑雅丽"]	2
大数据2203	["周佳佳"]	1

评分要点（5分）：

- 步骤一：总分、平均分计算正确 (1分)

- 步骤一：CASE WHEN 等级 4 级正确 (1分)

- 步骤一：`DENSE_RANK() OVER ()` 全体排名 (1分)

- 步骤一：`AVG() OVER ()` 计算全体均值差值 (1分)

- 步骤二：子查询过滤A级学生 + `COLLECT_LIST` 聚合 (1分)

评分注意事项

可接受的等价写法

以下写法在逻辑上等价，评分时均予以全分：

1. `CASE dept_id WHEN 10 THEN '技术部'` ⟺ `CASE WHEN dept_id = 10 THEN '技术部'`

2. `RANK()` ⇔ `DENSE_RANK()`（题目无明确要求时）

3. 使用子查询代替嵌套窗口函数（效果等同时）

4. `YEAR(order_date) = 2024` ⇔ `order_date LIKE '2024%'`

5. `IF(a >= b, ...)` ⇔ `CASE WHEN a >= b THEN ...`

常见扣分点

错误类型	扣分
GROUP BY 列与 SELECT 列不匹配	-1分
窗口函数中缺少 PARTITION BY	-1分
未使用子查询过滤排名结果	-1分
HAVING 与 WHERE 混淆	-1分
日期函数参数格式错误	-0.5分
列名拼写错误（结果正确）	-0.5分

关于执行效率

对于高分学生，可额外评价其 SQL 的执行效率：

- 避免重复计算（使用子查询或 CTE）

- 合理使用 LIMIT 减少全表扫描

- 分区键过滤减少 MapReduce 任务
