项目六 Hive 数据查询（DQL）综合实践工作手册

课程：Hive数据仓库应用

实践时间：约4小时

数据库：unit6db（路径：/user/hive/warehouse/unit6db）

数据文件目录：/usr/datas/unit6/

Beeline启动命令：`hive-bee`

MySQL信息：用户名 hiveuser，密码 123456

目录

章节	内容	建议时间
第一部分	环境准备与数据加载	25分钟
第二部分	SELECT基础查询（select_expr / ALL / DISTINCT）	30分钟
第三部分	WHERE子句与分区裁剪	25分钟
第四部分	GROUP BY 与 HAVING	30分钟
第五部分	LIMIT 与 SQL执行顺序	20分钟
第六部分	高阶排序（ORDER BY / CLUSTER BY / DISTRIBUTE BY+SORT BY）	35分钟
第七部分	UNION联合查询与子查询	30分钟
第八部分	CTE 公用表表达式	20分钟
第九部分	JOIN连接操作（6种）	45分钟
第十部分	综合挑战练习	40分钟

第一部分：环境准备与数据加载（25 分钟）

学习目标

- 创建实践数据库及所需表结构
- 将数据文件加载到Hive表中
- 掌握数据验证方法

1.1 准备数据文件

> **操作说明**：在终端（Terminal）中执行以下命令，创建数据目录并生成数据文件。

步骤1：创建数据目录

```
sudo mkdir -p /usr/datas/unit6
sudo chmod 777 /usr/datas/unit6
```

步骤2：创建美国新冠数据文件（us_covid19.txt）

```
cat > /usr/datas/unit6/us_covid19.txt << 'EOF'
2021-01-28,Autauga,Alabama,1001,5554,69
2021-01-28,Baldwin,Alabama,1003,17779,225
2021-01-28,Barbour,Alabama,1005,2083,42
2021-01-28,Bibb,Alabama,1007,2298,50
2021-01-28,Blount,Alabama,1009,5849,92
2021-01-28,Los Angeles,California,6037,1162766,17096
2021-01-28,San Diego,California,6073,210605,2340
2021-01-28,Orange,California,6059,225673,2538
2021-01-28,Riverside,California,6065,222296,2997
2021-01-28,San Bernardino,California,6071,258644,2734
2021-01-28,New York City,New York,36061,514561,29072
2021-01-28,Nassau,New York,36059,115703,3011
2021-01-28,Suffolk,New York,36103,125025,3280
2021-01-28,Erie,New York,36029,67888,1359
2021-01-28,Westchester,New York,36119,85698,1614
2021-01-28,Cook,Illinois,17031,328892,6637
```

```
2021-01-28,DuPage,Illinois,17043,61949,872
2021-01-28,Lake,Illinois,17097,52248,827
2021-01-28,Will,Illinois,17197,49891,631
2021-01-28,Maricopa,Arizona,4013,339843,7128
2021-01-28,Pima,Arizona,4019,73748,1471
2021-01-28,Pinal,Arizona,4021,35076,661
2021-01-28,Harris,Texas,48201,237567,3590
2021-01-28,Dallas,Texas,48113,186074,2399
2021-01-28,Tarrant,Texas,48439,149282,1727
2021-01-28,Bexar,Texas,48029,118094,1689
2021-01-28,Travis,Texas,48453,72638,725
2021-01-28,King,Washington,53033,60283,1091
2021-01-28,Pierce,Washington,53053,34481,426
2021-01-28,Snohomish,Washington,53061,29038,338
EOF
```

步骤3：创建学生数据文件（student.txt）

```
cat > /usr/datas/unit6/student.txt << 'EOF'
95001,李勇,男,20,CS
95002,刘晨,女,19,IS
95003,王敏,女,22,MA
95004,张立,男,19,IS
95005,刘刚,男,18,MA
95006,孙庆,男,23,CS
95007,易思玲,女,19,MA
95008,李娜,女,18,CS
95009,梦圆圆,女,18,MA
95010,孔小涛,男,19,CS
95011,包小柏,男,18,MA
95012,孙花,女,20,CS
95013,冯伟,男,21,CS
95014,王小丽,女,19,CS
95015,王君,男,18,MA
95016,钱国,男,21,MA
95017,王风娟,女,18,IS
95018,王一,女,19,IS
95019,邢小丽,女,19,IS
95020,赵钱,男,21,IS
95021,周二,男,17,MA
95022,郑明,男,20,MA
EOF
```

步骤4：创建员工数据文件（employee.txt）

```
cat > /usr/datas/unit6/employee.txt << 'EOF'
1201,gopal,manager,50000,TP
1202,manisha,cto,50000,TP
1203,khalil,dev,30000,AC
1204,prasanth,dev,30000,AC
1206,kranthi,admin,20000,TP
EOF
```

步骤5：创建员工住址数据文件（employee_address.txt）

```
cat > /usr/datas/unit6/employee_address.txt << 'EOF'
1201,288A,vgiri,jubilee
1202,108I,aoc,ny
1204,144Z,pgutta,hyd
1206,78B,old city,la
1207,720X,hitec,ny
EOF
```

步骤6：创建员工联系方式数据文件（employee_connection.txt）

```
cat > /usr/datas/unit6/employee_connection.txt << 'EOF'
1201,2356742,gopal@tp.com
1203,1661663,manisha@tp.com
1204,8887776,khalil@ac.com
1205,9988774,prasanth@ac.com
1206,1231231,kranthi@tp.com
EOF
```

【验证】 查看数据文件是否创建成功：

```
ls -la /usr/datas/unit6/
```

预期结果：应看到5个txt文件。

1.2 创建数据库与表结构

操作说明：启动 Beeline 并执行以下SQL语句。

启动 Beeline：

```
hive-bee
```

步骤1：创建并使用数据库

```
-- 创建数据库（如已存在则跳过）
CREATE DATABASE IF NOT EXISTS unit6db;
SHOW DATABASES;


-- 切换到unit6db数据库
USE unit6db;
SHOW TABLES;
```

📝 **记录结果**：当前数据库中已有哪些表?

答：_____

步骤2：创建美国新冠普通表 t_usa_covid19

```
-- 如表已存在，先删除
DROP TABLE IF EXISTS t_usa_covid19;


-- 创建普通表
CREATE TABLE t_usa_covid19 (
    count_date string,
    county     string,
    state      string,
    fips       int,
    cases      int,
    deaths     int
)
```

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';


SHOW TABLES;
```

步骤3：加载数据到 t_usa_covid19

```
LOAD DATA LOCAL INPATH '/usr/datas/unit6/us_covid19.txt'
INTO TABLE t_usa_covid19;


-- 验证数据加载
SELECT * FROM t_usa_covid19 LIMIT 10;
```

> 📝 **记录结果**：数据共有几行？
>
> 答：_____

步骤4：创建分区表 t_usa_covid19_p

```
-- 基于count_date日期和state州进行分区
CREATE TABLE IF NOT EXISTS t_usa_covid19_p (
    county string,
    fips   int,
    cases  int,
    deaths int
)
PARTITIONED BY (count_date string, state string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';


SHOW TABLES;
```

步骤5：将数据导入分区表（动态分区）

```
-- 开启动态分区模式
SET hive.exec.dynamic.partition.mode = nonstrict;


-- 动态分区插入
INSERT INTO TABLE t_usa_covid19_p
PARTITION (count_date, state)
SELECT county, fips, cases, deaths, count_date, state
FROM t_usa_covid19;


-- 验证分区数据
SELECT * FROM t_usa_covid19_p LIMIT 10;
SHOW PARTITIONS t_usa_covid19_p;
```

📝 **记录结果**：共有多少个分区？

答：_____

步骤6：创建学生表 student

```
DROP TABLE IF EXISTS student;


CREATE TABLE student (
    num  int,
    name string,
    sex  string,
    age  int,
    dept string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';


LOAD DATA LOCAL INPATH '/usr/datas/unit6/student.txt'
OVERWRITE INTO TABLE student;


SELECT * FROM student;
```

> 📝 **记录结果**：student表共有多少条记录?
>
> 答：_____

步骤7：创建员工相关表

```
-- 表1：员工基本信息表
DROP TABLE IF EXISTS employee;
CREATE TABLE employee (
    id     int,
    name   string,
    deg    string,
    salary int,
    dept   string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH '/usr/datas/unit6/employee.txt'
INTO TABLE employee;

-- 表2：员工住址信息表
DROP TABLE IF EXISTS employee_address;
CREATE TABLE employee_address (
    id     int,
    hno    string,
    street string,
    city   string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

LOAD DATA LOCAL INPATH '/usr/datas/unit6/employee_address.txt'
INTO TABLE employee_address;

-- 表3：员工联系方式表
DROP TABLE IF EXISTS employee_connection;
CREATE TABLE employee_connection (
    id    int,
    phno  string,
    email string
```

```
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';


LOAD DATA LOCAL INPATH '/usr/datas/unit6/employee_connection.txt'
INTO TABLE employee_connection;


-- 验证三张表数据
SELECT * FROM employee;
SELECT * FROM employee_address;
SELECT * FROM employee_connection;
```

📝 **思考题**：三张员工表的id字段存在哪些差异？这对JOIN操作有什么影响？

答：_____

✅ **第一部分完成检查清单**

- ☐ 5个数据文件已成功创建在 /usr/datas/unit6/ 目录
- ☐ t_usa_covid19 普通表创建并加载数据成功
- ☐ t_usa_covid19_p 分区表创建并加载数据成功
- ☐ student 表创建并加载数据成功
- ☐ employee / employee_address / employee_connection 三张表创建成功

第二部分：SELECT 基础查询（30 分钟）

学习目标

- 掌握 select_expr 的多种写法
- 理解 ALL 与 DISTINCT 的区别
- 能使用正则表达式选择列

2.1 select_expr 选择表达式

练习1：查询 t_usa_covid19_p 表中所有字段

```
-- 请填写完整的SQL语句
SELECT _____ FROM t_usa_covid19_p;
```

> 📝 **记录查询结果的行数**：_____

练习2：查询 t_usa_covid19_p 表中 county（县）、cases（确诊数）、deaths（死亡数）三个字段

```
-- 请填写完整的SQL语句
SELECT _____, _____, _____ FROM t_usa_covid19_p;
```

练习3：使用正则表达式查询所有以字母 'c' 开头的字段

> 💡 **提示**：需要先设置 `SET hive.support.quoted.identifiers = none;`，然后使用反引号+正则表达式

```
-- 第一步：设置参数
SET hive.support.quoted.identifiers = none;

-- 第二步：填写查询语句（查询所有c开头的列）
SELECT `^c.*` FROM t_usa_covid19_p LIMIT 5;
```

> 📝 **查询结果返回了哪些字段?**
>
> 答：_____

练习4：查询当前使用的数据库名称

```
-- 请填写完整的SQL语句（无需FROM子句）
SELECT _____;
```

> 📝 **结果是**：_____

练习5：统计 t_usa_covid19_p 表中 county（县）的总数量

```
SELECT count(county) FROM t_usa_covid19_p;
```

> 📝 **结果是**：_____

2.2 ALL 与 DISTINCT 去重

练习6：查询 t_usa_covid19_p 表中所有的 state（州），包含重复值

```
SELECT state FROM t_usa_covid19_p;
```

> 📝 **查询返回了多少行?** ：_____

练习7：使用 ALL 关键字查询（效果与练习6相同）

```
SELECT ALL state FROM t_usa_covid19_p;
```

> 📝 **两次查询结果行数是否一致？** 是 / 否

练习8：使用 DISTINCT 去重，查询有哪些不同的州

```
SELECT DISTINCT state FROM t_usa_covid19_p;
```

> 📝 **去重后共有多少个不同的州？**：_____

练习9：使用 DISTINCT 对多个字段组合去重（county + state）

```
SELECT DISTINCT county, state FROM t_usa_covid19_p;
```

> 📝 **思考题**：多字段DISTINCT的去重逻辑是什么？与单字段有何不同？
>
> 答：_____

练习10（扩展）：统计学生表中有多少个不同的系别（dept）

```
-- 请自行编写SQL语句
```

> 📝 **结果**：_____

✅ **第二部分小结**

语法	说明	示例
`SELECT *`	查询所有字段	
`SELECT col1, col2`	查询指定字段	
`` `SELECT ^c.*` ``	正则表达式选列	需先设置参数
`SELECT ALL`	返回全部行（含重复）	默认行为
`SELECT DISTINCT`	去除重复行	

第三部分：WHERE 子句与分区裁剪（25 分钟）

学习目标

- 掌握WHERE子句的各种条件表达式
- 理解为什么WHERE中不能使用聚合函数
- 掌握分区裁剪的概念和用法

3.1 WHERE 基础条件

练习1：布尔表达式测试——查询条件恒为false时的结果

```
SELECT * FROM t_usa_covid19_p WHERE 1 > 2;
```

> 📝 **查询返回多少行？为什么？**
>
> 答：_____

练习2：查询条件恒为true时的结果

```
SELECT * FROM t_usa_covid19_p WHERE 1 = 1 LIMIT 5;
```

> 📝 **这两个练习说明了WHERE子句的什么本质?**
>
> 答：_____

练习3：使用函数作为WHERE条件——找出州名字母长度超过10位的记录

```
SELECT * FROM t_usa_covid19_p
WHERE length(state) > 10;
```

> 📝 **符合条件的州有哪些?**
>
> 答：_____

练习4：WHERE子句中的子查询——找出county在某个特定列表中的记录

```
-- 查询county为'Los Angeles'或'Cook'的记录
SELECT * FROM t_usa_covid19
WHERE county IN ('Los Angeles', 'Cook');
```

> 📝 **结果返回了哪些记录?**
>
> 答：_____

练习5（错误示范）：尝试在WHERE子句中使用聚合函数（观察错误信息）

```
-- 注意：这条SQL会报错，请仔细阅读错误信息
SELECT state, count(deaths)
FROM t_usa_covid19_p
WHERE count(deaths) > 100
GROUP BY state;
```

📝 **记录错误信息的关键内容**：

答：_____

📝 **正确的写法应该是**（请将聚合条件改用HAVING）：

```sql
SELECT state, count(deaths)
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING _____;
```

3.2 分区裁剪（Partition Pruning）

💡 **概念说明**：分区裁剪是指在查询分区表时，通过WHERE子句中对分区字段的过滤，让Hive只扫描满足条件的分区，而不是全表扫描，从而提高查询效率。

练习6：对分区表进行全表扫描（无分区裁剪）

```
-- 查询所有加州(California)的记录，deaths > 1000
SELECT * FROM t_usa_covid19_p
WHERE deaths > 1000;
```

📝 **记录结果行数**：_____

练习7：使用单分区裁剪（state是分区字段）

```
-- state是分区字段，可以进行分区裁剪
SELECT * FROM t_usa_covid19_p
WHERE state = 'California'
AND deaths > 1000;
```

📝 **结果行数**：_____

📝 **与练习6相比，分区裁剪有什么优势?**

答：_____

练习8：多分区裁剪（同时指定 count_date 和 state 两个分区字段）

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
AND deaths > 1000;
```

📝 **结果行数**：_____

> **思考题**：哪些字段是分区字段？查询时应优先在WHERE中使用哪类字段？
>
> 答：_____

练习9（扩展）：在student表中，使用WHERE查询计算机系（CS）中年龄大于19岁的男生

```
-- 请自行编写SQL语句
```

> **符合条件的学生有哪些?**
>
> 答：_____

第四部分：GROUP BY 与 HAVING（30 分钟）

学习目标

- 理解GROUP BY的工作原理和语法限制
- 掌握各种聚合函数的使用
- 理解HAVING与WHERE的核心区别

4.1 GROUP BY 基础

练习1：按州(state)分组，统计每个州的县数量

```
SELECT state, count(county) AS county_count
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

📝 **记录查询结果**（填入表格）：

州名	县数量

练习2（错误示范）：将非聚合、非分组字段放入SELECT中（观察错误）

```
-- 这条SQL会报错：deaths不是分组字段，也没有被聚合函数处理
SELECT state, deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

📝 **记录错误信息的关键词**：_____

📝 **根据错误，说明GROUP BY的语法限制规则**：

答：_____

练习3：正确写法——将deaths用聚合函数处理

```
-- 按州分组，统计每个州的总死亡人数
SELECT state, sum(deaths) AS total_deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

📝 **死亡人数最多的州是哪个？**：_____

练习4：多聚合函数组合使用

```
-- 按州分组，同时计算：确诊总数、死亡总数、平均确诊数、最大确诊数
SELECT
    state,
    sum(cases)  AS total_cases,
    sum(deaths) AS total_deaths,
    round(avg(cases), 2) AS avg_cases,
    max(cases)  AS max_cases
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state;
```

📝 **round()函数的作用是什么?**

答：_____

练习5（扩展）：对student表，按性别和系别进行双字段分组，统计每组的人数和平均年龄

```
-- 请自行编写SQL语句
```

4.2 HAVING 子句

> 💡 **HAVING vs WHERE 对比**：
> - WHERE：在分组**之前**过滤行，不能使用聚合函数
> - HAVING：在分组**之后**过滤组，可以使用聚合函数

练习6（错误写法）：尝试在WHERE中使用聚合函数过滤分组后的结果

```
-- 这条SQL会报错
SELECT state, sum(deaths)
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND sum(deaths) > 10000
GROUP BY state;
```

> 📝 **此错误的原因是什么?**
>
> 答：_____

练习7：使用HAVING正确过滤——找出总死亡人数超过10000的州

```
SELECT state, sum(deaths) AS total_deaths
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING sum(deaths) > 10000;
```

> 📝 **符合条件的州有哪些?**
>
> 答：_____

练习8（优化写法）：使用别名在HAVING中引用聚合结果

```
-- 更高效的写法：having直接引用别名，避免重复计算
SELECT state, sum(deaths) AS cnts
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING cnts > 10000;
```

> 📝 **与练习7相比，此写法有什么优势?**
>
> 答：_____

练习9（综合）：找出计算机系(CS)或信息系(IS)中，平均年龄超过19岁且人数大于3人的系别

```
-- 请自行编写SQL语句，综合使用 WHERE + GROUP BY + HAVING
```

✅ WHERE 与 HAVING 对比总结

特性	WHERE	HAVING
执行时机	分组**之前**	分组**之后**
能否使用聚合函数	**不能**	**能**
过滤对象	行（row）	组（group）

是否可单独使用	是	一般配合GROUP BY

第五部分：LIMIT 与 SQL 执行顺序（20 分钟）

学习目标

- 掌握LIMIT子句的两种参数形式
- 理解并记忆SQL的完整执行顺序

5.1 LIMIT 子句

> 💡 **语法说明**：
> - `LIMIT n`：返回前n行
> - `LIMIT offset, n`：从第offset行开始（偏移量从0计），返回n行

练习1：查询加州（California）2021-01-28的所有记录（无限制）

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California";
```

> 📝 **共返回多少行？**：_____

练习2：LIMIT单参数——返回前5条记录

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
LIMIT 5;
```

练习3：LIMIT双参数——从第2条开始（偏移量为2），返回3条

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
LIMIT 2, 3;
```

📝 **注意**：第一个参数（2）是偏移量，从第几行开始（0表示第1行）？

答：_____

练习4（扩展）：查询student表中第3名到第5名的学生（按num排序后取第3~5条）

```
-- 提示：先用ORDER BY num排序，再用LIMIT取偏移
SELECT * FROM student
ORDER BY num
LIMIT _____, _____;
```

5.2 SQL 执行顺序分析

💡 **核心知识点**：Hive SQL的执行顺序（从先到后）：

FROM → WHERE → GROUP BY（含聚合函数）→ HAVING → SELECT → ORDER BY → LIMIT

练习5：分析以下SQL语句的执行步骤

```
SELECT state, sum(deaths) AS cnts
FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
GROUP BY state
HAVING cnts > 10000;
```

📝 **请按执行顺序分析每一步做了什么**：

1. FROM：_____
2. WHERE：_____
3. GROUP BY + sum()：_____
4. HAVING：_____
5. SELECT：_____

练习6（填空题）：根据执行顺序，判断以下哪些写法正确

```
（）A. SELECT state, sum(deaths) FROM t_usa_covid19_p WHERE sum(deaths) > 100 GROUP BY
state;
（）B. SELECT state, sum(deaths) FROM t_usa_covid19_p GROUP BY state HAVING sum(deaths) >
100;
（）C. SELECT state FROM t_usa_covid19_p GROUP BY state WHERE state = 'Texas';
（）D. SELECT state, count(*) AS cnt FROM t_usa_covid19_p GROUP BY state HAVING cnt > 3;
```

正确的选项是（多选）：_____

第六部分：高阶排序语法（35 分钟）

学习目标

- 掌握 ORDER BY（全局排序）的用法
- 理解 CLUSTER BY 的工作原理与局限性

- 掌握 DISTRIBUTE BY + SORT BY 的组合用法
- 理解四种排序方式的关系和区别

6.1 ORDER BY 全局排序

练习1：按deaths（死亡数）升序排序加州的记录

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
ORDER BY deaths;
-- 注意：默认ASC升序，NULL值排在最前面（NULLS FIRST）
```

练习2：按deaths降序排序，并限制返回前3条（死亡人数最多的3个县）

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
AND state = "California"
ORDER BY deaths DESC
LIMIT 3;
```

> 📝 **死亡人数最多的3个县是?**
>
> 答：_____

练习3：多字段排序——先按state升序，再按deaths降序

```
SELECT * FROM t_usa_covid19_p
WHERE count_date = "2021-01-28"
ORDER BY state ASC, deaths DESC
LIMIT 10;
```

练习4（扩展）：在student表中，按age升序排序（NULL值排最后）

```
SELECT * FROM student
ORDER BY age ASC NULLS LAST;
```

> 📝 **NULLS LAST的含义是什么?**
>
> 答：_____

6.2 CLUSTER BY 分组排序

> 💡 **概念**：CLUSTER BY 根据指定字段进行哈希分组，并在每个组内对该字段进行**升序**排序（只能升序）。分组数量取决于reducetask的数量。

练习5：不指定reduce task个数，对student表按num进行CLUSTER BY

```
-- 默认不指定reduce task个数
SELECT * FROM student CLUSTER BY num;
```

> 📝 **日志中提示了什么信息（Number of reduce tasks...）?**
>
> 答：_____

练习6：手动设置2个reduce task，再次执行CLUSTER BY

```
SET mapreduce.job.reduces = 2;
SELECT * FROM student CLUSTER BY num;
```

> 📝 **设置2个reducetask后，结果分成了几组?**
>
> 答：_____

练习7（理解局限性）：尝试用CLUSTER BY实现"按sex分组，组内按age降序"——验证是否可行

```
-- 尝试这种写法（会报错）
SELECT * FROM student CLUSTER BY sex ORDER BY age DESC;
```

> 📝 **报错信息是什么？为什么CLUSTER BY无法实现此需求?**
>
> 答：_____

6.3 DISTRIBUTE BY + SORT BY

> 💡 **概念**:
> - DISTRIBUTE BY：负责按指定字段进行哈希**分组**（分区）
> - SORT BY：负责每个分组内的**局部排序**
> - 两者字段**可以不同**，弥补了CLUSTER BY的局限性

练习8：按sex性别分组，组内按age年龄**降序**排列

```
SELECT * FROM student
DISTRIBUTE BY sex
SORT BY age DESC;
```

> 📝 **观察结果：男生组和女生组是否分开了？每组内年龄是否降序？**
>
> 答：_____

练习9：验证等价关系——当DISTRIBUTE BY和SORT BY字段相同时，等价于CLUSTER BY

```
-- 方式1：DISTRIBUTE BY + SORT BY（字段相同）
SELECT * FROM student DISTRIBUTE BY num SORT BY num LIMIT 5;

-- 方式2：CLUSTER BY
SELECT * FROM student CLUSTER BY num LIMIT 5;
```

> 📝 **两种方式的结果是否相同？这说明了什么？**
>
> 答：_____

✅ 四种排序方式对比总结

排序方式	Reducer数量	分组	排序范围	是否支持降序
ORDER BY	1个	无	全局	✅
SORT BY	多个	无	组内局部	✅
DISTRIBUTE BY	多个	✅	无	—
CLUSTER BY	多个	✅	组内局部	❌（只能升序）

第七部分：UNION 联合查询与子查询（30 分钟）

学习目标

- 掌握UNION和UNION ALL的区别
- 理解FROM子句中的子查询
- 掌握WHERE子句中的不相关子查询（IN/NOT IN）和相关子查询（EXISTS）

7.1 UNION 联合查询

练习1：创建两个测试表，理解UNION与UNION ALL的差异

```
-- 创建测试表A
DROP TABLE IF EXISTS t_union_a;
CREATE TABLE t_union_a (id int, name string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';


-- 创建测试表B
DROP TABLE IF EXISTS t_union_b;
CREATE TABLE t_union_b (id int, name string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';


-- 插入测试数据（表A中有重复数据）
INSERT INTO t_union_a VALUES (1, 'Alice'), (2, 'Bob'), (3, 'Charlie');
INSERT INTO t_union_b VALUES (3, 'Charlie'), (4, 'David'), (5, 'Eve');
```

练习2：使用 UNION ALL（保留重复行）

```
SELECT id, name FROM t_union_a
UNION ALL
SELECT id, name FROM t_union_b;
```

> 📝 **结果共几行？**：_____（两表行数之和）

练习3：使用 UNION（自动去重）

```
SELECT id, name FROM t_union_a
UNION
SELECT id, name FROM t_union_b;
```

> 📝 **结果共几行？UNION与UNION ALL的区别是什么？**
>
> 答：_____

练习4：UNION在实际场景中的应用——合并不同州的数据

```
-- 合并California和Texas的新冠数据
SELECT county, cases, deaths, state FROM t_usa_covid19_p
WHERE state = 'California'
UNION ALL
SELECT county, cases, deaths, state FROM t_usa_covid19_p
WHERE state = 'Texas';
```

> 📝 **结果共几行？**：_____

7.2 FROM 子句中的子查询

> 💡 **概念**：将一个SELECT查询的结果作为临时表，放在FROM子句中使用。子查询必须有别名。

练习5：使用FROM子查询——先获取student表中num和name，再从结果中过滤

```
-- 外层查询引用子查询结果
SELECT num
```

```
FROM (
    SELECT num, name FROM student
) tmp
WHERE tmp.name LIKE '王%';
```

> 📝 **姓王的同学学号有哪些?**
>
> 答: _____

练习6：FROM子查询结合UNION——找出各州确诊数最多的记录

```
SELECT state, max_cases
FROM (
    SELECT state, max(cases) AS max_cases
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-28"
    GROUP BY state
) t
WHERE max_cases > 100000;
```

> 📝 **结果中有哪些州?**
>
> 答: _____

7.3 WHERE 子句中的子查询

练习7：不相关子查询（IN）——查询employee_address表中，id在employee表中存在的记录

```
-- 不相关子查询：子查询结果作为IN的列表
SELECT *
FROM employee_address
WHERE id IN (SELECT id FROM employee);
```

> 📝 **结果返回几行？哪些id有匹配？**
>
> 答：_____

练习8：不相关子查询（NOT IN）——查询employee_address表中，id不在employee表中的记录

```
SELECT *
FROM employee_address
WHERE id NOT IN (SELECT id FROM employee);
```

> 📝 **哪些员工地址记录没有对应的员工基本信息？**
>
> 答：_____

练习9：相关子查询（EXISTS）——查询存在联系方式记录的员工信息

```
-- EXISTS相关子查询：子查询引用了父查询的列
SELECT e.id, e.name
FROM employee e
WHERE EXISTS (
    SELECT ec.id
    FROM employee_connection ec
    WHERE e.id = ec.id
);
```

> 📝 **结果有哪些员工？**
>
> 答：_____

第八部分：CTE 公用表表达式（20 分钟）

学习目标

- 理解CTE的概念和使用场景
- 掌握WITH...AS语法
- 能使用多个CTE组合复杂查询

> 💡 **CTE概念**：公用表表达式（Common Table Expression）是一个命名的临时结果集，在单个SQL语句的执行范围内有效。使用WITH子句定义，紧接在SELECT/INSERT之前。

练习1：使用单个CTE——获取死亡人数大于1000的加州县

```
WITH high_deaths AS (
    SELECT county, cases, deaths
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-28"
    AND state = "California"
    AND deaths > 1000
)
SELECT * FROM high_deaths
ORDER BY deaths DESC;
```

> 📝 **共有多少个县符合条件?**
>
> 答：_____

练习2：使用CTE计算各州统计指标，再筛选

```
WITH state_stats AS (
```

```
    SELECT
        state,
        sum(cases)  AS total_cases,
        sum(deaths) AS total_deaths,
        count(county) AS county_count
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-28"
    GROUP BY state
)
SELECT state, total_cases, total_deaths
FROM state_stats
WHERE total_deaths > 5000
ORDER BY total_deaths DESC;
```

📝 **结果有哪些州?**

答: _____

练习3：使用多个CTE——分别计算高确诊县和高死亡县，然后关联

```
WITH high_cases AS (
    SELECT county, state, cases
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-28"
    AND cases > 100000
),
high_deaths AS (
    SELECT county, state, deaths
    FROM t_usa_covid19_p
    WHERE count_date = "2021-01-28"
    AND deaths > 2000
)
SELECT hc.county, hc.state, hc.cases, hd.deaths
FROM high_cases hc
INNER JOIN high_deaths hd
ON hc.county = hd.county AND hc.state = hd.state;
```

📝 **同时满足高确诊和高死亡条件的县有哪些?**

> 答：_____

练习4（扩展）：使用CTE重写第四部分练习9的SQL（找出平均年龄超过19岁且人数大于3人的系别）

```
-- 请用WITH...AS语法改写：
```

✅ CTE与子查询的比较

特性	CTE（WITH...AS）	子查询（Subquery）
可读性	更高，逻辑清晰	嵌套层次多时较难读
是否可复用	同语句内可多次引用	不可复用
执行范围	单个SQL语句内	单个子查询
适用场景	复杂多步骤查询	简单过滤

第九部分：JOIN 连接操作（45 分钟）

学习目标

- 掌握Hive支持的6种JOIN类型
- 理解各种JOIN的工作原理和返回结果差异
- 能根据业务需求选择合适的JOIN类型

💡 **JOIN概念回顾**：JOIN用于根据两个或多个表中列之间的关系，从这些表中联合查询数据。

Hive 3.1.2支持的6种JOIN：

1. INNER JOIN（内连接）
2. LEFT JOIN（左外连接）
3. RIGHT JOIN（右外连接）
4. FULL OUTER JOIN（全外连接）
5. LEFT SEMI JOIN（左半开连接）
6. CROSS JOIN（交叉连接/笛卡尔积）

查看三张员工表的数据（确认ID的对应关系）

```
SELECT '=== employee ===' AS info;
SELECT * FROM employee;


SELECT '=== employee_address ===' AS info;
SELECT * FROM employee_address;


SELECT '=== employee_connection ===' AS info;
SELECT * FROM employee_connection;
```

📝 **分析三张表的ID交集情况**：

- employee 的 id：_____
- employee_address 的 id：_____
- employee_connection 的 id：_____
- 三表共同拥有的 id：_____

9.1 INNER JOIN 内连接

> 💡 **规则**：只返回两个表中**都有匹配**的行。

练习1：查询同时有员工信息和住址信息的员工

```
-- inner join（inner可以省略）
SELECT e.id, e.name, e_a.city, e_a.street
FROM employee e
INNER JOIN employee_address e_a
ON e.id = e_a.id;
```

> 📝 **结果共几行？哪些员工出现在结果中？**
>
> 答：_____

练习2：三表INNER JOIN——查询同时有住址和联系方式的员工完整信息

```
SELECT e.id, e.name, e_a.city, e_conn.phno, e_conn.email
FROM employee e
INNER JOIN employee_address e_a ON e.id = e_a.id
INNER JOIN employee_connection e_conn ON e.id = e_conn.id;
```

> 📝 **三表JOIN后，结果共几行？**
>
> 答：_____

9.2 LEFT JOIN 左外连接

> 💡 **规则**：以左表为基准，返回左表所有行；右表匹配上的显示数据，匹配不上的显示NULL。

练习3：查询所有员工及其联系方式（没有联系方式的员工也要显示）

```
SELECT e.id, e.name, e_conn.phno, e_conn.email
FROM employee e
LEFT JOIN employee_connection e_conn
ON e.id = e_conn.id;
```

> 📝 **结果共几行？哪个员工的联系方式显示为NULL？**
>
> 答：_____

练习4：LEFT JOIN的实际应用——统计每个员工有无住址记录

```
SELECT e.id, e.name,
       CASE WHEN e_a.id IS NULL THEN '无住址记录' ELSE e_a.city END AS address_status
FROM employee e
LEFT JOIN employee_address e_a ON e.id = e_a.id;
```

9.3 RIGHT JOIN 右外连接

> 💡 **规则**：以右表为基准，返回右表所有行；左表匹配不上的显示NULL。

练习5：以employee_connection为右表，查询所有联系方式及对应员工信息

```
SELECT e.id, e.name, e_conn.phno, e_conn.email
FROM employee e
RIGHT JOIN employee_connection e_conn
ON e.id = e_conn.id;
```

📝 **哪条联系方式记录没有对应的员工基本信息（显示NULL）？**

答：_____

📝 **思考题**：如何用LEFT JOIN改写以上RIGHT JOIN查询（互换表的位置）？

```sql
-- 请改写：
SELECT e.id, e.name, e_conn.phno, e_conn.email
FROM employee_connection e_conn
_____ JOIN employee e
ON _____;
```

9.4 FULL OUTER JOIN 全外连接

💡 **规则**：返回左表和右表的**所有行**，无论是否有匹配。不匹配的一侧显示NULL。

练习6：查询employee和employee_address的完整关联情况

```
SELECT e.id, e.name, e_a.city, e_a.street
FROM employee e
FULL OUTER JOIN employee_address e_a
ON e.id = e_a.id;
```

📝 **结果共几行？哪些行出现了NULL值？**

答：_____

练习7（思考）：填写下表，对比INNER、LEFT、RIGHT、FULL OUTER JOIN的结果行数

JOIN类型	左表(employee)行数	右表(employee_address)行数	结果行数
INNER JOIN	5	5	
LEFT JOIN	5	5	
RIGHT JOIN	5	5	
FULL OUTER JOIN	5	5	

9.5 LEFT SEMI JOIN 左半开连接

> 💡 **规则**：返回左表中，在右表中有匹配的行（类似EXISTS子查询），**不返回右表的任何列。**

练习8：查询有住址记录的员工信息（使用LEFT SEMI JOIN）

```
SELECT *
FROM employee e
LEFT SEMI JOIN employee_address e_addr
ON e.id = e_addr.id;
```

> 📝 **结果共几行？与INNER JOIN的结果有何不同？**
>
> 答：_____

练习9（对比）：用INNER JOIN实现相同效果，并对比结果

```
SELECT e.*
FROM employee e
INNER JOIN employee_address e_addr
ON e.id = e_addr.id;
```

📝 **LEFT SEMI JOIN 与 INNER JOIN 的关键区别是什么?**

答：_____

9.6 CROSS JOIN 交叉连接（笛卡尔积）

⚠️ **注意**：CROSS JOIN会产生笛卡尔积（两表行数的乘积），数据量大时慎用!

练习10：对employee和employee_address进行CROSS JOIN

```
SELECT a.id, a.name, b.id AS addr_id, b.city
FROM employee a
CROSS JOIN employee_address b
LIMIT 10;
```

📝 **employee(5行) × employee_address(5行) = 多少行?**

答：_____

练习11：CROSS JOIN加WHERE条件过滤（等价于INNER JOIN）

```
-- 下面两个查询结果是否相同？
-- 方式1：CROSS JOIN + WHERE
```

```
SELECT a.id, a.name, b.city
FROM employee a
CROSS JOIN employee_address b
WHERE a.id = b.id;


-- 方式2：INNER JOIN
SELECT a.id, a.name, b.city
FROM employee a
INNER JOIN employee_address b
ON a.id = b.id;
```

📝 **两种方式结果是否相同？：_____**

✅ 六种JOIN类型综合对比

JOIN类型	返回左表无匹配行	返回右表无匹配行	返回右表列	特点
INNER JOIN	❌	❌	✅	只返回双方都匹配的行
LEFT JOIN	✅ (NULL)	❌	✅	左表全部保留
RIGHT JOIN	❌	✅ (NULL)	✅	右表全部保留
FULL OUTER	✅ (NULL)	✅ (NULL)	✅	两表全部保留
LEFT SEMI	❌	❌	❌	左表在右表中存在即返回
CROSS JOIN	—	—	✅	笛卡尔积，慎用

第十部分：综合挑战练习（40 分钟）

学习目标

- 综合运用本章所有知识点

- 解决真实业务场景问题

> 💡 **提示**：以下题目可能需要综合使用 WHERE、GROUP BY、HAVING、JOIN、子查询、CTE等多种技术，请灵活组合。

挑战1：统计每个州的新冠数据概况

要求：查询2021-01-28每个州的县数量、总确诊数、总死亡数、平均确诊数（保留2位小数），并按总确诊数降序排列。

```
-- 请独立编写SQL语句：
```

> 📝 **确诊数最多的3个州是?**
>
> 答：_____

挑战2：找出高风险县（确诊数大于各州平均确诊数的县）

要求：使用CTE或子查询，找出每个州内确诊数超过该州平均确诊数的县。

```
-- 提示：先用CTE计算每州平均确诊数，再JOIN筛选
```

```

```

挑战3：学生表多维度分析

① 统计各系男女生人数及平均年龄（按系别和性别分组）

```
-- 请编写SQL：
```

② 找出人数最多的系别（使用子查询或CTE）

```
-- 请编写SQL：
```

③ 查询比计算机系(CS)平均年龄大的其他系学生

```
-- 请编写SQL：
```

挑战4：员工信息完整性分析

要求：查询所有员工，显示其基本信息、住址城市（无住址显示"未知"）、联系邮箱（无邮箱显示"未留"）。

```
-- 提示：使用FULL OUTER JOIN或多个LEFT JOIN
-- 提示：使用 COALESCE() 或 CASE WHEN 处理NULL值
```

挑战5：NEW YORK州县级数据分析

使用以下步骤完成分析：

1. 使用CTE获取New York州的所有县数据
2. 计算死亡率（deaths/cases × 100）
3. 找出死亡率最高的3个县
4. 使用ORDER BY和LIMIT输出结果

```
WITH ny_data AS (
    -- 第1步：请填写CTE内容


)
-- 第2、3、4步：请在此编写主查询
```

自我评估清单

完成本次实践后，请对以下知识点进行自评（✅已掌握 / ⚠️基本掌握 / ❌需复习）：

知识点	自评

SELECT基础语法（select_expr、正则选列）	
ALL与DISTINCT的区别	
WHERE子句的用法和限制	
分区表与分区裁剪	
GROUP BY的语法规则	
聚合函数（sum/count/avg/max/min）	
HAVING与WHERE的区别	
LIMIT的单参数和双参数用法	
SQL执行顺序	
ORDER BY全局排序	
CLUSTER BY的原理与局限	
DISTRIBUTE BY + SORT BY	
UNION与UNION ALL	
FROM子句子查询	
WHERE子句子查询（IN/EXISTS）	
CTE（WITH...AS）语法	
INNER JOIN	
LEFT / RIGHT JOIN	
FULL OUTER JOIN	
LEFT SEMI JOIN	
CROSS JOIN	

📌 实践总结

今天遇到的主要问题：

1. _____

2. _____
3. _____

解决方法：

1. _____
2. _____
3. _____

心得体会：

_____

_____

《Hive数据仓库应用教程》— 项目六综合实践工作手册