结合您的电信数据仓库和 BI 开发运维经验，快速掌握 dbt 和 Airflow 的学习路径：

📊 核心概念快速对应表

传统电信数据仓库概念	dbt 对应概念	Airflow 对应概念
ETL/ELT 流程	dbt 负责 Transform 层	Airflow 负责调度和编排
存储过程/脚本	dbt 模型（SQL + Jinja）	DAG 中的任务
作业调度器（如 Control-M）	dbt Cloud 调度	Airflow DAG 调度
数据质量检查	dbt Tests	Airflow 数据质量监控
依赖关系管理	DAG（有向无环图）	DAG 任务依赖

🚀 7 天速成学习计划

第 1-2 天：dbt 核心概念

1. **dbt 基础（4 小时）**
    - dbt 在 Modern Data Stack 中的位置
    - 与您熟悉的 Kimball/Inmon 建模结合
    - 安装 dbt Core：`pip install dbt`
2. **关键概念映射（4 小时）**

```yaml
# 您的现有知识 → dbt 概念
- 维度表 → dbt 模型（materialized='table'）
- 事实表 → dbt 模型 + 增量加载
- ETL 脚本 → dbt 模型 + Jinja 模板
- 调度脚本 → dbt 运行命令
```

3. **实战练习**：将您现有的一个 SQL 转换脚本改写成 dbt 模型

第 3-4 天：dbt 高级功能

1. **dbt 项目结构（2 小时）**

```text
telecom_dw_project/
```

```
├── models/
│   ├── staging/      # 对应 ODS 层
│   ├── marts/        # 对应数据集市
│   └── intermediate/ # 中间层
├── tests/            # 数据测试
└── dbt_project.yml   # 项目配置
```

2. **电信行业实用模式（4 小时）**

```sql
-- 示例：客户维度增量模型
{{ config(materialized='incremental') }}

SELECT
    customer_id,
    msisdn,
    segment,
    activation_date,
    {{ current_timestamp() }} as etl_load_time
FROM {{ ref('stg_customer') }}
WHERE activation_date >= (SELECT MAX(activation_date) FROM {{ this }})
```

3. **数据测试（2 小时）**

```yaml
# 电信数据质量测试
models:
  - name: fact_cdr
    tests:
      - not_null: [call_id, msisdn]
      - unique: call_id
      - accepted_values:
          field: call_duration
          values: ['>= 0']
```

第 5-6 天：Airflow 核心概念

1. **Airflow 架构（3 小时）**
    - Web Server, Scheduler, Executor
    - 与您熟悉的调度系统对比
2. **第一个电信 DAG（4 小时）**

```python
from airflow import DAG
```

```
from airflow.operators.bash import BashOperator
from datetime import datetime

default_args = {
    'owner': 'telecom_bi',
    'retries': 3,
}

with DAG(
    'telecom_cdr_pipeline',
    default_args=default_args,
    schedule_interval='0 2 * * *',  # 每天凌晨 2 点
    start_date=datetime(2024, 1, 1),
) as dag:

    extract_task = BashOperator(
        task_id='extract_cdr',
        bash_command='python /scripts/extract_cdr.py'
    )

    dbt_run = BashOperator(
        task_id='transform_with_dbt',
```

```
    bash_command='cd /dbt_project && dbt run --models fact_cdr+'
)

dbt_test = BashOperator(
    task_id='data_quality_check',
    bash_command='cd /dbt_project && dbt test'
)

extract_task >> dbt_run >> dbt_test
```

3. **实操部署（3 小时）**

```bash
# 快速启动 Airflow
docker-compose up -d

# 部署 dbt 项目
dbt init telecom_dw
```

第 7 天：整合与优化

1. **dbt + Airflow 最佳实践（4 小时）**

- 使用 dbt Cloud + Airflow 的混合架构
- 错误处理与重试机制
- 监控与告警配置

2. **电信场景实战（4 小时）**
   - 构建 CDR（通话详单）处理管道
   - KPI 计算自动化
   - 数据血缘和文档自动化

重点突破策略

1. 利用现有知识迁移

- 将已有的 SQL 脚本逐步迁移到 dbt
- 将现有的调度作业映射到 Airflow DAGs
- 复用已有的数据质量检查规则

2. 优先掌握的核心功能

dbt：

- 模型物化策略（table/view/incremental）
- Jinja 模板和宏
- 包管理和版本控制
- 文档生成

Airflow：

- DAG 定义和调度
- 任务依赖管理
- 变量和连接管理
- 监控和告警

3. 实用学习资源

- **dbt 官方文档**：重点看 Quickstart 和模型配置
- **Airflow Tutorial**：官方入门教程
- **GitHub 模板**：
  - dbt-labs/jaffle-shop（dbt 示例）
  - astronomer/airflow-guides（Airflow 示例）

🔗 快速实践项目建议

创建一个**电信用户行为分析**的最小可行项目：

1. 使用 dbt 构建用户 360 度视图
2. 使用 Airflow 调度每日 ETL
3. 实现数据质量监控
4. 生成自动化报表

💡 经验复用技巧

您的优势在于：

1. **数据建模经验** → 直接应用于 dbt 模型设计
2. **ETL 开发经验** → 快速理解 dbt 转换逻辑
3. **运维经验** → 轻松掌握 Airflow 的部署和监控
4. **电信领域知识** → 设计符合行业的测试和质量检查

关键建议：不要试图一次性学会所有功能，先构建一个端到端的简单管道，然后逐步添加复杂功能。

需要更具体的电信数据场景示例或遇到特定问题时，我可以提供针对性指导！