Instruções para Execução do DAG de ETL com Apache Airflow

Este documento fornece as instruções para a configuração e execução de um DAG de ETL utilizando o Apache Airflow. O DAG extrai dados de arquivos, os transforma e os carrega em um formato de sua escolha.

# Código do DAG:

import pandas as pd  
import pendulum  
from airflow import DAG  
from airflow.operators.python import PythonOperator  
from datetime import timedelta  
from airflow.models import TaskInstance  
  
# Definindo a função de extração de dados  
def extract():  
 try:  
 print("Extraindo dados...")  
 # Caminhos dos arquivos  
  
 # Caminhos dos arquivos no contêiner Docker  
 caminho\_oportunidades = "/opt/airflow/dags/database/registros\_oportunidades.json"  
 caminho\_sellout = "/opt/airflow/dags/database/sellout.parquet"  
  
 # Carregar os dados dos arquivos  
 oportunidades = pd.read\_json(caminho\_oportunidades)  
 sellout = pd.read\_parquet(caminho\_sellout)  
   
 print("Dados carregados com sucesso.")  
   
 return oportunidades, sellout  
   
 except Exception as e:  
 raise Exception(f"Erro na etapa de extração: {e}")  
  
# Definindo a função de transformação de dados  
def transform():  
 try:  
 print("Transformando dados...")  
 # Aqui você colocaria a lógica de transformação  
 print("Transformações concluídas.")  
   
 except Exception as e:  
 raise Exception(f"Erro na etapa de transformação: {e}")  
  
# Definindo a função de carregamento de dados  
def load():  
 try:  
 print("Carregando dados...")  
 # Aqui você colocaria a lógica para carregar os dados (por exemplo, para um arquivo Excel)  
 print("Dados carregados com sucesso.")  
   
 except Exception as e:  
 raise Exception(f"Erro na etapa de carregamento: {e}")  
  
# Função para verificar o estado de retry da task  
def check\_task\_retry\_status(task\_instance: TaskInstance):  
 # Alterado de 'up\_for\_retry' para 'ready\_for\_retry'  
 if task\_instance.ready\_for\_retry:  
 print(f"Tarefa {task\_instance.task\_id} está esperando para ser reexecutada.")  
 else:  
 print(f"Tarefa {task\_instance.task\_id} não está mais aguardando reexecução.")  
  
# Definindo a DAG  
default\_args = {   
 'owner': 'airflow',  
 'retries': 3, # Tenta até 3 vezes  
 'retry\_delay': timedelta(minutes=5), # A cada 5 minutos  
}  
  
dag = DAG(  
 'etl\_airflow\_final', # Nome da DAG  
 default\_args=default\_args,  
 description='ETL com Airflow',  
 schedule=None, # Substituindo schedule\_interval por schedule  
 start\_date=pendulum.today('UTC').add(days=-1), # Substituindo days\_ago por pendulum  
 catchup=False,  
)  
  
# Definindo as tasks  
extract\_task = PythonOperator(  
 task\_id='extract\_data',  
 python\_callable=extract,  
 dag=dag,  
)  
  
transform\_task = PythonOperator(  
 task\_id='transform\_data',  
 python\_callable=transform,  
 dag=dag,  
)  
  
load\_task = PythonOperator(  
 task\_id='load\_data',  
 python\_callable=load,  
 dag=dag,  
)  
  
# Função que será chamada após a execução da task para checar o estado de retry  
def check\_retry(\*\*kwargs):  
 # O Airflow já passa 'execution\_date' no contexto  
 execution\_date = kwargs['execution\_date'] # Usando o execution\_date da execução atual  
 task\_instance = TaskInstance(task=extract\_task, execution\_date=execution\_date) # Passando a task da extração  
 check\_task\_retry\_status(task\_instance)  
  
# Adicionando a tarefa de verificação de retry após as outras  
check\_retry\_task = PythonOperator(  
 task\_id='check\_retry\_status',  
 python\_callable=check\_retry,  
 provide\_context=True, # Necessário para passar o execution\_date  
 dag=dag,  
)  
  
# Definindo a ordem de execução  
extract\_task >> transform\_task >> load\_task >> check\_retry\_task

# Instruções para Execução:

1. Configuração do ambiente:  
 - Certifique-se de ter o Apache Airflow instalado. Você pode instalar via pip:  
 pip install apache-airflow  
  
2. Estrutura de arquivos:  
 - O arquivo .json e .parquet precisam estar no caminho "/opt/airflow/dags/database/". Caso esteja utilizando um ambiente diferente, altere os caminhos dos arquivos na função extract() para o diretório correspondente.  
  
3. Executar o Airflow:  
 - Inicialize o Airflow no seu terminal:  
 airflow db init # Inicializa o banco de dados do Airflow  
 airflow webserver --port 8080 # Inicia o servidor web do Airflow  
 airflow scheduler # Inicia o agendador  
  
4. Subir a DAG:  
 - Coloque o arquivo com o código acima na pasta "dags" do seu diretório do Airflow.  
 - Acesse o Airflow pelo navegador, normalmente no http://localhost:8080/, e veja a DAG na interface gráfica.  
  
5. Executar a DAG:  
 - Na interface do Airflow, você pode acionar a execução manualmente clicando em "Trigger DAG" para rodar a DAG de ETL.