from airflow.operators.python import BranchPythonOperator from airflow.operators.python import PythonOperator from airflow.operators.dummy import DummyOperator from airflow.models import Variable

def analyser$_{q}ualite_{d}onnees(**context)$: """$Analyselaqualitédesdonnéesetchoisitlebontraitement$""" $Simulationd'analysedequalitéexecution_{d}ate = context['execution_{d}ate']fichier_{s}ource = f"/data/raw/execution_{d}ate.strftime('$

Métriques de qualité (simulées) metriques$_{q}ualite = 'completude' : 0.85, 85'exactitude' : 0.92, 92'consist$

print(f" Métriques de qualité : metriques$_{q}ualite$")

Seuils de qualité configurables seuil$_{e}leve = float(Variable.get("seuil_{q}ualite_{e}leve", 0.9))seuil_{m}oyen = float(Variable.get("seuil_{q}ualite_{m}oyen", 0.7))$

Décision basée sur la qualité if metriques$_{q}ualite['completude'] >= seuil_{e}leveandmetriques_{q}ualite['exac$ $seuil_{e}leve : return"traitement_{a}utomatique"elifmetriques_{q}ualite['completude'] >= seuil_{m}oyen : return"traitement_{a}vec_{n}ettoyage"else : return"traitement_{m}anuel_{r}evision"$

except Exception as e : print(f" Erreur lors de l'analyse : e") return "traitement$_{e}rreur$"

def choisir$_{m}ethode_{a}ggregation(**context)$: """$Choisitlaméthoded'agrégationselonlevolumededonn$ $context['ti'].xcom_{p}ull(task_{i}ds =' analyser_{q}ualite_{d}onnees')$

volume$_{d}onnees = 500000Simulation$

if volume$_{d}onnees > 1000000 : return"aggregation_{d}istribuee"elifvolume_{d}onnees > 100000 : return"aggregation_{m}emoire"else : return"aggregation_{s}imple"$

Tâches de décision decision$_{q}ualite = BranchPythonOperator(task_{i}d = "analyser_{q}ualite_{d}onnees", py$ $analyser_{q}ualite_{d}onnees)$

decision$_{a}ggregation = BranchPythonOperator(task_{i}d = "choisir_{m}ethode_{a}ggregation", python_{c}alla$ $choisir_{m}ethode_{a}ggregation)$

Tâches de traitement selon la qualité traitement$_{a}uto = PythonOperator(task_{i}d = "traitement_{a}utomat$ $lambda : print("Traitementautomatique - hautequalité"))$

traitement$_{n}ettoyage = PythonOperator(task_{i}d = "traitement_{a}vec_{n}ettoyage", python_{c}allable = lambda : print("Traitementavecnettoyage - qualitémoyenne"))$

traitement$_{m}anuel = PythonOperator(task_{i}d = "traitement_{m}anuel_{r}evision", python_{c}allable = lambda : print("Traitementmanuel - qualitéfaible"))$

traitement$_{e}rreur = PythonOperator(task_{i}d = "traitement_{e}rreur", python_{c}allable = lambda : print("Traitementd'erreur - donnéesproblématiques"))$

Tâches d'agrégation agg$_{d}istribuee = PythonOperator(task_{i}d = "aggregation_{d}istribuee", python_{c}alla$ $lambda : print("Agrégationdistribuée(Spark)"))$

agg$_{m}emoire = PythonOperator(task_{i}d = "aggregation_{m}emoire", python_{c}allable = lambda : print("Agrégationenmémoire(Pandas)"))$

agg$_{s}imple = PythonOperator(task_{i}d = "aggregation_{s}imple", python_{c}allable = lambda : print("Agrégationsimple(SQL)"))$

Tâche finale finalisation = PythonOperator(task$_{i}d = "finaliser_{t}raitement", python_{c}allable = lambda : print("Traitementfinaliséavecsuccès"))$

DÉPENDANCES COMPLEXES Premier niveau : décision qualité decision$_{q}ualite >> [traitement_{a}uto, traitement_{n}ettoyage, traitement_{m}anuel, traitement_{e}rreur]$

Deuxième niveau : décision agrégation (sauf pour erreur) [traitement$_auto, traitement_nettoyage, traiteme decision_aggregation$

Troisième niveau : agrégation decision$_aggregation >> [agg_distribuee, agg_memoire, agg_simple]$

Niveau final : toutes les branches convergent [agg$_distribuee, agg_memoire, agg_simple, traitement_erreur] finalisation$

