**Отчёт о выполнении домашнего задания к семинару номер семь**

Файл d7dag.py

import os  
import logging  
import pandas as pd  
from airflow.operators.bash import BashOperator  
from sqlalchemy import create\_engine  
from sqlalchemy.exc import SQLAlchemyError  
from airflow import DAG  
from airflow.operators.python import PythonOperator  
from datetime import datetime, timedelta  
import pendulum  
  
# Настройки логирования  
logging.basicConfig(level=logging.INFO)  
logger = logging.getLogger(\_\_name\_\_)  
  
default\_args = {  
 'owner': 'Kostia',  
 'depends\_on\_past': False,  
 'start\_date': pendulum.datetime(2024, 6, 1, tz='Europe/Moscow'),  
 'email': ['dom@dom.ru'],  
 'email\_on\_failure': False,  
 'email\_on\_retry': False,  
 'retries': 0,  
 'retry\_delay': timedelta(minutes=5)  
}  
  
dag2 = DAG('Kostia002',  
 default\_args=default\_args,  
 description="seminar\_7",  
 catchup=False,  
 schedule\_interval='0 8 \* \* \*')  
  
  
def percent(\*\*kwargs):  
 files = ['/opt/airflow/dags/d4\_1.xlsx', '/opt/airflow/dags/d4\_2.xlsx', '/opt/airflow/dags/d4\_3.xlsx']  
  
 # Настройки подключения к MySQL  
 MYSQL\_USER = "airflow"  
 MYSQL\_PASSWORD = "airflow"  
 MYSQL\_HOST = "mysql-db"  
 MYSQL\_DB = "spark"  
  
 # Подключение к MySQL через SQLAlchemy с таймаутом  
 con = create\_engine(f"mysql+pymysql://{MYSQL\_USER}:{MYSQL\_PASSWORD}@{MYSQL\_HOST}:3306/{MYSQL\_DB}",  
 connect\_args={'connect\_timeout': 10})  
  
 for file in files:  
 if not os.path.exists(file):  
 logger.error(f"❌ Файл {file} не найден!")  
 raise FileNotFoundError(f"Файл {file} не найден!")  
  
 logger.info(f"📂 Обработка файла {file}")  
 df = pd.read\_excel(file, engine='openpyxl')  
  
 required\_columns = ['Платеж по основному долгу', 'Платеж по процентам']  
 for col in required\_columns:  
 if col not in df.columns:  
 logger.error(f"❌ В файле {file} отсутствует колонка {col}!")  
 raise KeyError(f"Ошибка: в файле {file} отсутствует колонка {col}!")  
  
 df['долг'] = df['Платеж по основному долгу'].cumsum().round(2)  
 df['проценты'] = df['Платеж по процентам'].cumsum().round(2)  
  
 # Проверка существования таблицы перед заменой  
 try:  
 table\_exists = con.dialect.has\_table(con.connect(), "credit", schema="spark")  
 if not table\_exists or os.path.basename(file) == "d4\_1.xlsx":  
 df.to\_sql('credit', con, schema='spark', if\_exists='replace', index=False, chunksize=500)  
 else:  
 df.to\_sql('credit', con, schema='spark', if\_exists='append', index=False, chunksize=500)  
 except SQLAlchemyError as e:  
 logger.error(f"Ошибка при записи в БД: {e}")  
 raise  
  
  
# Устанавливаем Python-библиотеки  
task1 = BashOperator(  
 task\_id='pip\_install',  
 bash\_command="""  
 pip install --no-cache-dir cryptography pandas pymysql sqlalchemy openpyxl pendulum  
 """,  
 dag=dag2  
)  
  
# Этот DAG автоматизирует обработку платежных данных из Excel-файлов  
task2 = PythonOperator(  
 task\_id='python3',  
 dag=dag2,  
 python\_callable=percent  
)  
  
task1 >> task2 # Устанавливаем порядок выполнения

Лог выполнения  


