360522 - Anderson Costa da Silva

Exercise 1

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime, timedelta

def extrair_dados():
    # Simula extração de dados
    print("Extraindo dados da fonte...")
    return {'compras': 200, 'artefatos': 200}

def transformar_dados(**context):
    dados = context['ti'].xcom_pull(task_ids='extrair')
    # Processa os dados
    dados['total'] = dados['compras'] + dados['artefatos']
    return dados

def carregar_dados(**context):
    dados = context['ti'].xcom_pull(task_ids='transformar')
    print(f"Carregando dados: {dados}")

with DAG(
    'DAG_ANDERSON',
    default_args={'retries': 2},
    start_date=datetime(2024, 1, 1),
    schedule_interval='*/2 * * * *',
    catchup=False
) as dag:

    t1 = PythonOperator(task_id='extrair', python_callable=extrair_dados)
    t2 = PythonOperator(task_id='transformar',
python_callable=transformar_dados)
    t3 = PythonOperator(task_id='carregar',
python_callable=carregar_dados)

    t1 >> t2 >> t3
```

Exercise 2

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime, timedelta
import pandas as pd
import numpy as np
```

```
# Default arguments for the DAG
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2024, 1, 1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

# Initialize the DAG
dag = DAG(
    'ANDERSON_EX2',
    default_args=default_args,
    description='A simple DAG that creates and transforms synthetic data',
    schedule_interval=timedelta(days=3),
    catchup=False,
)

def create_synthetic_data(**context):
    """Create synthetic customer data"""
    np.random.seed(42)
    n_records = 100

    data = {
        'customer_id': range(1, n_records + 1),
        'age': np.random.randint(18, 80, n_records),
        'purchase_amount': np.random.uniform(10, 500, n_records).round(2),
        'items_purchased': np.random.randint(1, 20, n_records),
        'region': np.random.choice(['North', 'South', 'East', 'West'],
n_records),
        'customer_name': np.random.choice(['Fulano', 'Cicrano',
'Beltrano'], n_records),
    }

    df = pd.DataFrame(data)

    # Push to XCom for next task
    context['ti'].xcom_push(key='raw_data', value=df.to_json())
    print(f"Created {len(df)} synthetic records")

def transform_data(**context):
    """Transform the data using pandas"""
    # Pull data from previous task
    raw_data_json = context['ti'].xcom_pull(key='raw_data',
task_ids='create_data')
    df = pd.read_json(raw_data_json)

    # Add calculated columns
    df['price_per_item'] = (df['purchase_amount'] /
df['items_purchased']).round(2)
    df['customer_segment'] = pd.cut(df['age'],
                                    bins=[0, 30, 50, 100],
```

```
                                        labels=['Young', 'Middle', 'Senior'])

    # Filter high-value customers
    df['is_high_value'] = df['purchase_amount'] > 200

    # Push transformed data
    context['ti'].xcom_push(key='transformed_data', value=df.to_json())
    print(f"Transformed data with {len(df)} records")

def analyze_data(**context):
    """Analyze the transformed data"""
    transformed_json = context['ti'].xcom_pull(key='transformed_data',
task_ids='transform_data')
    df = pd.read_json(transformed_json)

    # Calculate summary statistics
    summary = {
        'total_customers': len(df),
        'avg_purchase_amount': df['purchase_amount'].mean().round(2),
        'high_value_customers': df['is_high_value'].sum(),
        'customers_by_region': df['region'].value_counts().to_dict(),
    }

    print("Analysis Summary:")
    for key, value in summary.items():
        print(f"{key}: {value}")
    df.to_csv('/tmp/bd100_anderson_2.csv')

# Define tasks
create_data_task = PythonOperator(
    task_id='create_data',
    python_callable=create_synthetic_data,
    dag=dag,
)

transform_data_task = PythonOperator(
    task_id='transform_data',
    python_callable=transform_data,
    dag=dag,
)

analyze_data_task = PythonOperator(
    task_id='analyze_data',
    python_callable=analyze_data,
    dag=dag,
)

# Set task dependencies
create_data_task >> transform_data_task >> analyze_data_task
```

Exercise 3

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime
import pandas as pd
#from pymongo import MongoClient
import json

def read_csv(**context):
    # Read CSV file
    df = pd.read_csv('/opt/airflow/bd1.csv')

    # Select first 5 rows and specific columns
    data = df[['customer_id', 'age', 'purchase_amount']].head(5)

    # Convert to list of dictionaries
    records = data.to_dict('records')

    # Push to XCom for other tasks to use
    context['ti'].xcom_push(key='records', value=records)
    print(f"Read {len(records)} records from CSV")

def insert_row(row_index, **context):
    # Pull data from XCom
    records = context['ti'].xcom_pull(task_ids='read_csv_task',
key='records')
    record = records[row_index]
    with open(str(row_index)+'.json', 'w') as f:
        json.dump(record, f)


# Define DAG
with DAG(
    dag_id='ANDERSON_EX3',
    start_date=datetime(2024, 1, 1),
    schedule_interval=None,
    catchup=False,
    tags=['csv', 'mongodb', 'parallel']
) as dag:

    # Task 1: Read CSV
    read_task = PythonOperator(
        task_id='read_csv_task',
        python_callable=read_csv
    )

    # Tasks 2-11: Insert each row
    insert_tasks = []
    for i in range(10):
        insert_task = PythonOperator(
            task_id=f'insert_row_{i}',
            python_callable=insert_row,
            op_kwargs={'row_index': i}
```

```
            )
            insert_tasks.append(insert_task)

        # Set dependencies: read first, then all inserts in parallel
        read_task >> insert_tasks
```