airflow\dags\kaggle_dag.py

```python
# 685.652, Spring 2025 - Group 6 Final Project
# kaggle_dag.py

# This DAG gets billboard hot 100 chart data from kaggle
# Cleans and transforms it
# Then loads it into postgres

from airflow import DAG
from airflow.operators.empty import EmptyOperator
from airflow.providers.postgres.operators.postgres import PostgresOperator
from airflow.operators.python_operator import PythonOperator
from airflow.models import Variable
from airflow.providers.postgres.hooks.postgres import PostgresHook
from datetime import datetime, timedelta
import os
import zipfile
import subprocess
import pandas as pd
import uuid

default_args = {
    'owner': 'group6',
    '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),
}


# Retrieves a specific file from kaggle via API
def get_kag_file():

    # User needs to set up account on kaggle and generate API key
    # Set the KAG_USERNAME and KAG_KEY in variables.json
    os.environ['KAGGLE_USERNAME'] = Variable.get("KAG_USERNAME")
    os.environ['KAGGLE_KEY'] = Variable.get("KAG_KEY")

    # Retrieving a specific dataset
    dataset_name = 'elizabethearhart/billboard-hot-1001958-2024'
    zip_file_name = f"{dataset_name.split('/')[-1]}.zip"

    # Data is to go into data subdirectory
    data_dir = os.path.join(os.getcwd(), 'data')
    os.makedirs(data_dir, exist_ok=True)

    # Download the dataset by running the kaggle command
    # The kaggle package gets installed on build
    subprocess.run([
        'kaggle', 'datasets', 'download',
        '-d', dataset_name,
        '--force', '-p', data_dir])

    # Unzip to data subdirectory
    zip_path = os.path.join(data_dir, zip_file_name)
    try:

        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall(data_dir)
            extracted_files = zip_ref.namelist()

        if extracted_files:
            extracted_file_name = extracted_files[0]
            print(f"Extracted Kaggle file: {extracted_file_name} to data directory")

    except PermissionError:
        print("Error extracting ZIP file: Permission denied")
        print("Close Kaggle data csv if you already have it open in another program")
        return None

    # Extract first file
    extracted_file_name = extracted_files[0] if extracted_files else None
    print(f"Extracted Kaggle file: {extracted_file_name} to data directory")

    # Keep extracted file, delete zip
    os.remove(zip_path)
    print(f"The ZIP file {zip_file_name} has been deleted\n")

    return extracted_file_name


# Processes kaggle file in data directory and returns a dataframe
def process_kag_file(file_name):
    print(f"\nProcessing Kaggle file: {file_name} ...\n"
          "File contains Bilboard Hot 100 chart data from 1958 to 2024\n"
          "It contains - chart_week, title, performer, current_week, last_week, peak_pos,
wks_on_chart\n"
          "Duplicate song title, performer pairs will be removed\n")

    data_dir = os.path.join(os.getcwd(), 'data')
    os.makedirs(data_dir, exist_ok=True)
    kaggle_file_path = os.path.join(data_dir, file_name)

    # Read kaggle file from data directory
    df = pd.read_csv(kaggle_file_path, encoding='utf-8')

    # Strip and lowercase string cols
    df['title'] = df['title'].str.strip().str.lower()
    df['performer'] = df['performer'].str.strip().str.lower()

    # Force chart_wk to date for sorting
    df['chart_week'] = pd.to_datetime(df['chart_week'], errors='coerce')

    # Replace empty strings with None
    df.replace('', None, inplace=True)

    df_kaggle = df.copy()

    # Replace some strings to help with matching to Spotify data
    df_kaggle['performer'] = df_kaggle['performer'].str.replace(r' \(featuring ([^)]*)\)', r',\1', regex=True)
    df_kaggle['performer'] = df_kaggle['performer'].str.replace(r' \(feat. ([^)]*)\)', r',\1', regex=True)
    df_kaggle['performer'] = df_kaggle['performer'].str.replace('featuring', ',')
    df_kaggle['performer'] = df_kaggle['performer'].str.replace('feat.', ',')
    df_kaggle['performer'] = df_kaggle['performer'].str.replace('&', 'and')
    df_kaggle['performer'] = df_kaggle['performer'].str.replace(r' ,', r',', regex=True)
    df_kaggle['title'] = df_kaggle['title'].str.replace(r' \(featuring.*$', '', regex=True)

    df_kaggle['top_artist'] = df_kaggle['performer'].str.split(',').str[0]
    df_kaggle['performer'] = df_kaggle['performer'].str.split(',')

    # Force int columns to numeric, force any NaNs to None
    int_columns = ['current_week', 'last_week', 'peak_pos', 'wks_on_chart']
    for col in int_columns:
        df_kaggle[col] = pd.to_numeric(df_kaggle[col], errors='coerce')
    df_kaggle[int_columns] = df_kaggle[int_columns].where(pd.notnull(df_kaggle[int_columns]), None)

    # Force int columns to Int64 (needed for one column that was being read as float)
    # Replace 0 with None
    for col in int_columns:
        df_kaggle[col] = df_kaggle[col].astype('Int64')
    for col in int_columns:
        df_kaggle[col] = df_kaggle[col].replace(0, None)


    print(f"Number of Kaggle dataset rows before removing duplicates: {len(df_kaggle)}")

    # Group by top_artist and title, then aggregate
    date_groups = df_kaggle.groupby(['top_artist', 'title']).agg(
        performer=('performer', 'first'),           # Keep the first performer string
        chart_week_min=('chart_week', 'min'),      # Rename agg results directly
        chart_week_max=('chart_week', 'max'),
        current_week=('current_week', 'last'),
        peak_pos=('peak_pos', 'min'),
        wks_on_chart=('wks_on_chart', 'max')
    ).reset_index() # Makes 'top_artist' and 'title' columns again

    # Rename the columns after aggregation
    date_groups.rename(columns={
        'performer': 'artists',            # Now rename the carried-over performer
        'title': 'song_name',              # Rename title
        'chart_week_min': 'wk_first_charted',
        'chart_week_max': 'wk_last_charted',
        'current_week': 'last_chart_pos',
        'peak_pos': 'peak_chart_pos',
        'wks_on_chart': 'total_wks_on_chart'
    }, inplace=True)

    # Assign the correctly aggregated and renamed DataFrame back
    df_kaggle = date_groups

    print(f"Number of Kaggle dataset rows after removing duplicates: {len(df_kaggle)}\n")

    # This column reordering should now work
    df_kaggle = df_kaggle[['top_artist', 'artists', 'song_name', 'peak_chart_pos',
                           'last_chart_pos', 'total_wks_on_chart',
                           'wk_first_charted', 'wk_last_charted']]


    # Write to CSV for record
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    unique_hot100 = f"unique_hot100_{timestamp}.csv"
    kag_file_name = os.path.join(data_dir, unique_hot100)

    df_kaggle.to_csv(kag_file_name, index=False,
                    encoding='utf-8-sig')
    print(f"Kaggle dataset processed and saved to {unique_hot100}\n")

    count = (df_kaggle['peak_chart_pos'] == 1).sum()
    print(f"Processed Kaggle dataset contains {len(df_kaggle)} songs in total and "
          f"{count} songs that peaked at #1")
    print(f"Dataframe has columns: {', '.join(df_kaggle.columns)}\n")

    return df_kaggle



# Loads kaggle data into postgres
def load_billboard_data():
    kag_name = get_kag_file()
    df_kaggle = process_kag_file(kag_name)

    # get postgres connection
    pg_hook = PostgresHook(postgres_conn_id='pg_group6')
    with pg_hook.get_conn() as conn:
        with conn.cursor() as cur:
            for index, row in df_kaggle.iterrows():
                group6_id = uuid.uuid5(uuid.NAMESPACE_DNS, str(row['song_name'].strip() +
row['top_artist'].strip()))

                cur.execute("""INSERT INTO billboard_chart_data (group6_id, top_artist,
artists, song_name, peak_chart_pos,
                            last_chart_pos, total_wks_on_chart, wk_first_charted,
wk_last_charted)
                            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)""",
                            (group6_id,
                            row['top_artist'],
                            row['artists'],
                            row['song_name'],
                            row['peak_chart_pos'],
                            row['last_chart_pos'],
                            row['total_wks_on_chart'],
                            row['wk_first_charted'],
                            row['wk_last_charted']))


with DAG(
        'kaggle_dag',
        default_args=default_args,
        description='Dag for kaggle billboard data',
        schedule_interval=None,
        catchup=False
    ) as kaggle_dag:

    start_task = EmptyOperator(task_id='start')

    create_tables = PostgresOperator(
        task_id='create_tables',
        postgres_conn_id='pg_group6',
        sql='sql/billboard_create.sql'
    )

    load_billboard_data = PythonOperator(
        task_id='load_billboard_data',
        python_callable=load_billboard_data
    )

    end_task = EmptyOperator(task_id='end')

    start_task >> create_tables >> load_billboard_data >> end_task

```