

airflow\dags\matching_dag OPTIONAL.py

```
1 # 685.652, Spring 2025 - Group 6 Final Project
2 # matching_dag.py
3
4 # Matches up closely related tracks
5 # From different datasets
6 # Using fuzzy string matching
7 # And overrides group6_id based on findings
8
9
10
11 from airflow import DAG
12 from airflow.operators.empty import EmptyOperator
13 from airflow.providers.postgres.operators.postgres import PostgresOperator
14 from airflow.operators.python_operator import PythonOperator, BranchPythonOperator
15 from airflow.models import Variable
16 from airflow.providers.postgres.hooks.postgres import PostgresHook
17 import pandas as pd
18 from rapidfuzz import process, fuzz
19 import numpy as np
20 import time
21 from datetime import datetime, timedelta
22 from math import sqrt
23 import os
24 import uuid
25
26
27 default_args = {
28     'owner': 'group6',
29     'depends_on_past': False,
30     'start_date': datetime(2025, 1, 1),
31     'email_on_failure': False,
32     'email_on_retry': False,
33     'retries': 1,
34     'retry_delay': timedelta(minutes=5),
35 }
36
37
38 # Bring data from all existing tables in
39 def load_all_from_db():
40     pg_hook = PostgresHook(postgres_conn_id='pg_group6')
41
42     with pg_hook.get_conn() as conn:
43         with conn.cursor() as cur:
44             cur.execute("SELECT * FROM spotify_tracks")
45             spot_data = cur.fetchall()
46             spot_columns = [desc[0] for desc in cur.description]
47
48             cur.execute("SELECT * FROM billboard_chart_data")
```

```

49     kag_data = cur.fetchall()
50     kag_columns = [desc[0] for desc in cur.description]
51
52     cur.execute("SELECT * FROM lastfm_tracks")
53     lastfm_data = cur.fetchall()
54     lastfm_columns = [desc[0] for desc in cur.description]
55
56     cur.execute("SELECT * FROM acousticbrainz_features")
57     ab_data = cur.fetchall()
58     ab_columns = [desc[0] for desc in cur.description]
59
60     df_spot = pd.DataFrame(spot_data, columns=spot_columns)
61     df_kag = pd.DataFrame(kag_data, columns=kag_columns)
62     df_lastfm = pd.DataFrame(lastfm_data, columns=lastfm_columns)
63     df_ab = pd.DataFrame(ab_data, columns=ab_columns)
64
65     return df_spot, df_kag, df_lastfm, df_ab
66
67 # Helper for match functions
68 def append_match_result(results, spot_artists, spot_song_name, other_artists,
69                         other_song_name,
70                         match_found, other_index, spot_index, match_score,
71                         artist_match_pct, name_match_pct):
72     results.append({
73         'spot_artists': spot_artists,
74         'spot_song_name': spot_song_name,
75         'other_artists': other_artists,
76         'other_song_name': other_song_name,
77         'match_found': match_found,
78         'other_index': other_index,
79         'spot_index': spot_index,
80         'best_match_score': match_score,
81         'best_match_artist_match_pct': artist_match_pct,
82         'best_match_name_match_pct': name_match_pct
83     })
84
85
86 # Match spotify tracks to kaggle tracks
87 # Overwrite group6_id based on findings
88 def match_spot_to_kag(df_spot, df_kag):
89
90     # Store the original dataframe before filtering
91     df_kag_original = df_kag.copy()
92
93     # To decrease matching time, filter out songs
94     # That didn't reach a certain chart position
95     # Because fuzzy string matching is slow
96     KAG_FILTER = 15
97     df_kag_filtered = df_kag[df_kag['peak_chart_pos'] <= KAG_FILTER]

```

```

98     df_kag_filtered = df_kag_filtered.reset_index() # Keep original index in 'index' column
99
100    print(f"Filtered Kaggle dataset to {len(df_kag_filtered)} records with peak chart
101      position <= {KAG_FILTER}")
102    print(f"Will search for matches within the {len(df_spot)} Spotify records")
103    print(f"This may take a while...")
104
105    # Will be used to track matches
106    df_kag_filtered['has_spot_match'] = False
107    df_kag_filtered['spot_match_index'] = None
108    df_kag_filtered['exact_match'] = False
109    df_spot['has_kag_match'] = False
110    df_spot['kag_match_index'] = None
111
112    results = []
113
114    start_time = time.time()
115
116    # Pre-compute data for faster matching
117    kag_performers = set(df_kag_filtered['top_artist'].unique())
118    performer_indices = {performer: df_kag_filtered[df_kag_filtered['top_artist'] == performer].index[0]
119                          for performer in kag_performers}
120
121    performer_to_titles = {}
122    for idx, row in df_kag_filtered.iterrows():
123        performer = row['top_artist']
124        title = row['song_name']
125        if performer not in performer_to_titles:
126            performer_to_titles[performer] = set()
127            performer_to_titles[performer].add(title)
128
129    kag_performer_title_to_index = {}
130    for idx, row in df_kag_filtered.iterrows():
131        performer = row['top_artist']
132        title = row['song_name']
133        kag_performer_title_to_index[(performer, title)] = idx
134
135    # Pre-compute word sets
136    kag_artist_words_dict = {idx: set(row['top_artist'].split()) for idx, row in
137    df_kag_filtered.iterrows()}
138    kag_title_words_dict = {idx: set(row['song_name'].split()) for idx, row in
139    df_kag_filtered.iterrows()}
140
141    # Pre-compute Spotify artist and title word sets
142    spot_artist_words_dict = {idx: set(row['top_artist'].split()) for idx, row in
143    df_spot.iterrows()}
144    spot_title_words_dict = {idx: set(row['song_name'].split()) for idx, row in
145    df_spot.iterrows()}
146
147    # Create artist+first letter of title index

```

```
143     artist_title_first = {}
144     for idx, row in df_kag_filtered.iterrows():
145         if row['song_name']:
146             key = (row['top_artist'], row['song_name'][0])
147             if key not in artist_title_first:
148                 artist_title_first[key] = []
149                 artist_title_first[key].append(idx)
150
151     # Iterate through each spotify track
152     for index, row in df_spot.iterrows():
153
154         # If there is already a match, skip
155         if df_spot.at[index, 'has_kag_match']:
156             continue
157
158         artist = row['top_artist']
159         song_name = row['song_name']
160         match_found = False
161
162         # Get pre-computed word sets for this Spotify track
163         spot_artist_words = spot_artist_words_dict[index]
164         spot_title_words = spot_title_words_dict[index]
165
166         best_match_performer = None
167         best_match_title = None
168         best_match_score = 0
169         best_match_artist_match_pct = 0
170         best_match_name_match_pct = 0
171         best_match_kag_index = None
172
173         # Print progress every 10 records
174         if index % 10 == 0:
175             print(f"Processing record {index + 1} of {len(df_spot)}...")
176
177         if artist in kag_performers:
178             curr_performer = artist
179
180             # If the song name is in the performer's title set,
181             # Then it is an exact match
182             # These will already have the same group6_id
183             if song_name in performer_to_titles.get(curr_performer, set()):
184                 filtered_kag_index = kag_performer_title_to_index.get((curr_performer,
185 song_name))
186
187                 # Get the original index from the filtered dataframe
188                 original_kag_index = df_kag_filtered.at[filtered_kag_index, 'index']
189                 best_match_kag_index = filtered_kag_index
190
191                 # Update both dataframes
192                 df_spot.at[index, 'has_kag_match'] = True
193                 df_spot.at[index, 'kag_match_index'] = original_kag_index
```

```

192     df_kag_filtered.at[filtered_kag_index, 'has_spot_match'] = True
193     df_kag_filtered.at[filtered_kag_index, 'spot_match_index'] = index
194     df_kag_filtered.at[filtered_kag_index, 'exact_match'] = True
195
196     match_found = True
197     append_match_result(results, artist, song_name, curr_performer, song_name,
198                         match_found, original_kag_index, index, 100, 100, 100)
199     continue
200
201 # Start from that artist's first appearance in the dataset
202 # For better likelihood of quick match
203 start_index = performer_indices[curr_performer]
204 for i in range(start_index, len(df_kag_filtered)):
205
206     # If there is already a match, skip
207     if df_kag_filtered.at[i, 'has_spot_match']:
208         continue
209
210     # Get the current top artist and title
211     curr_performer = df_kag_filtered.iloc[i]['top_artist']
212     title = df_kag_filtered.iloc[i]['song_name']
213
214     # If the first letter of the artist or title doesn't match, skip
215     if curr_performer[0] != artist[0]:
216         continue
217     if title[0] != song_name[0]:
218         continue
219
220     # If the artist words don't intersect, skip
221     curr_artist_words = kag_artist_words_dict[i]
222     if not spot_artist_words.intersection(curr_artist_words):
223         continue
224
225     # If the title words don't intersect, skip
226     curr_title_words = kag_title_words_dict[i]
227     if not spot_title_words.intersection(curr_title_words):
228         continue
229
230     # Check QRatio fuzzy match for artist and title
231     # Skip if either is very low
232     artist_score = fuzz.QRatio(artist, curr_performer)
233     if artist_score < 50:
234         continue
235     title_score = fuzz.QRatio(song_name, title)
236     if title_score < 50:
237         continue
238
239     # Geometric mean is good - penalizes one very low score more
240     geo_mean_score = sqrt(artist_score * title_score)
241

```

```

242     # If the fuzzy match is better than the current best, update
243     # This block was mostly for testing
244     # To find the right threshold for a good match
245     if geo_mean_score > best_match_score:
246         best_match_performer = curr_performer
247         best_match_title = title
248         best_match_score = geo_mean_score
249         best_match_artist_match_pct = artist_score
250         best_match_name_match_pct = title_score
251         best_match_kag_index = i
252
253     # If the fuzzy match is close enough,
254     # Then it is a match
255     if geo_mean_score >= 70:
256         match_found = True
257
258         filtered_kag_index = i
259         original_kag_index = df_kag_filtered.at[filtered_kag_index, 'index']
260
261         df_spot.at[index, 'has_kag_match'] = True
262         df_spot.at[index, 'kag_match_index'] = original_kag_index
263         df_kag_filtered.at[filtered_kag_index, 'has_spot_match'] = True
264         df_kag_filtered.at[filtered_kag_index, 'spot_match_index'] = index
265
266         append_match_result(results, artist, song_name, curr_performer, title,
267                             match_found, original_kag_index, index,
268                             best_match_score,
269                             best_match_artist_match_pct, best_match_name_matc-
270                             h_pct)
271         break
272
273     # Need to wrap back around and keep looking
274     # Back to the start index
275     # All else in this block is the same
276     if not match_found:
277         for i in range(0, start_index):
278             if df_kag_filtered.at[i, 'has_spot_match']:
279                 continue
280
281                 curr_performer = df_kag_filtered.iloc[i]['top_artist']
282                 title = df_kag_filtered.iloc[i]['song_name']
283
284                 if curr_performer[0] != artist[0]:
285                     continue
286                 if title[0] != song_name[0]:
287                     continue
288
289                 curr_artist_words = kag_artist_words_dict[i]
290                 if not spot_artist_words.intersection(curr_artist_words):
291                     continue

```

```
290
291     curr_title_words = kag_title_words_dict[i]
292     if not spot_title_words.intersection(curr_title_words):
293         continue
294
295     artist_score = fuzz.QRatio(artist, curr_performer)
296     if artist_score < 50:
297         continue # Skip early
298
299     # Only calculate title score if artist score is promising
300     title_score = fuzz.QRatio(song_name, title)
301     if title_score < 50:
302         continue # Skip early
303
304     geo_mean_score = sqrt(artist_score * title_score)
305
306     if geo_mean_score > best_match_score:
307         best_match_performer = curr_performer
308         best_match_title = title
309         best_match_score = geo_mean_score
310         best_match_artist_match_pct = artist_score
311         best_match_name_match_pct = title_score
312         best_match_kag_index = i
313
314     if geo_mean_score >= 70:
315         match_found = True
316
317         filtered_kag_index = i
318         original_kag_index = df_kag_filtered.at[filtered_kag_index, 'index']
319
320         df_spot.at[index, 'has_kag_match'] = True
321         df_spot.at[index, 'kag_match_index'] = original_kag_index
322         df_kag_filtered.at[filtered_kag_index, 'has_spot_match'] = True
323         df_kag_filtered.at[filtered_kag_index, 'spot_match_index'] = index
324
325         append_match_result(results, artist, song_name, curr_performer,
326                             title,
327                             best_match_score,
328                             best_match_name_match_pct)
329         break
330
331     # This is all the same as above
332     # Just for artists that don't have an exact match between datasets
333     else:
334         for i in range(0, len(df_kag_filtered)):
335             if df_kag_filtered.at[i, 'has_spot_match']:
336                 continue
337
338             curr_performer = df_kag_filtered.iloc[i]['top_artist']
```

```

338     title = df_kag_filtered.iloc[i]['song_name']
339
340     if curr_performer[0] != artist[0]:
341         continue
342     if title[0] != song_name[0]:
343         continue
344
345     curr_artist_words = kag_artist_words_dict[i]
346     if not spot_artist_words.intersection(curr_artist_words):
347         continue
348
349     curr_title_words = kag_title_words_dict[i]
350     if not spot_title_words.intersection(curr_title_words):
351         continue
352
353     artist_score = fuzz.QRatio(artist, curr_performer)
354     if artist_score < 50:
355         continue
356     title_score = fuzz.QRatio(song_name, title)
357     if title_score < 50:
358         continue
359     geo_mean_score = sqrt(artist_score * title_score)
360
361     if geo_mean_score > best_match_score:
362         best_match_performer = curr_performer
363         best_match_title = title
364         best_match_score = geo_mean_score
365         best_match_artist_match_pct = artist_score
366         best_match_name_match_pct = title_score
367         best_match_kag_index = i
368
369     if geo_mean_score >= 70:
370         match_found = True
371
372         filtered_kag_index = i
373         original_kag_index = df_kag_filtered.at[filtered_kag_index, 'index']
374
375         df_spot.at[index, 'has_kag_match'] = True
376         df_spot.at[index, 'kag_match_index'] = original_kag_index
377         df_kag_filtered.at[filtered_kag_index, 'has_spot_match'] = True
378         df_kag_filtered.at[filtered_kag_index, 'spot_match_index'] = index
379
380         append_match_result(results, artist, song_name, curr_performer, title,
381                             match_found, original_kag_index, index,
382                             best_match_score,
383                             best_match_artist_match_pct, best_match_name_matc-
384                             h_pct)
385         break
386
387     if not match_found and best_match_kag_index is not None:

```

```

386         original_kag_index = df_kag_filtered.at[best_match_kag_index, 'index'] if
387         best_match_kag_index is not None else None
388         append_match_result(results, artist, song_name, best_match_performer,
389         best_match_title,
390                         match_found, original_kag_index, index, best_match_score,
391                         best_match_artist_match_pct, best_match_name_match_pct)
390     elif not match_found:
391         append_match_result(results, artist, song_name, best_match_performer,
392         best_match_title,
393                         match_found, best_match_kag_index, index, best_match_score,
394                         best_match_artist_match_pct, best_match_name_match_pct)
394
395     # After all matches are found,
396     # Update the original dataframe with the match information, but preserve existing matches
397     for index, row in df_spot.iterrows():
398         if row['has_kag_match'] and pd.notna(row['kag_match_index']):
399             original_kag_index = int(row['kag_match_index'])
400             if 0 <= original_kag_index < len(df_kag): # Validate index
401                 # Only update if the group6_ids are different
402                 if df_kag.at[original_kag_index, 'group6_id'] != row['group6_id']:
403                     df_kag.at[original_kag_index, 'group6_id'] = row['group6_id']
404
405             end_time = time.time()
406             elapsed_time = end_time - start_time
407             print(f"\nTotal time taken for Spotify and Kaggle matching: {elapsed_time:.2f} seconds")
408
409
410     # Write to CSV for viewing results
411     results_df = pd.DataFrame(results)
412     results_df.sort_values(by=[ 'best_match_score'], ascending=False, inplace=True)
413     results_df.reset_index(drop=True, inplace=True)
414
415     timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
416     match_results_file_name = f"spottokaggle_match_results_{timestamp}.csv"
417     data_dir = os.path.join(os.getcwd(), 'data')
418     os.makedirs(data_dir, exist_ok=True)
419     match_results_file_path = os.path.join(data_dir, match_results_file_name)
420     results_df.to_csv(match_results_file_path, index=False)
421
422     print(f"Successfully wrote to {match_results_file_name}\n")
423
424     return df_spot, df_kag
425
426
427
428
429 # All the same as above
430 # Just between spotify and lastfm
431 def match_spot_to_lastfm(df_spot, df_lastfm):
432
433     start_time = time.time()

```

```

434
435     # Initialize tracking columns
436     df_lastfm['has_spot_match'] = False
437     df_lastfm['spot_match_index'] = None
438     df_lastfm['exact_match'] = False
439     df_spot['has_lastfm_match'] = False
440     df_spot['lastfm_match_index'] = None
441
442     results = []
443
444     # Pre-compute word sets for faster matching
445     lastfm_artists = set(df_lastfm['artist'].unique())
446     artist_indices = {artist: df_lastfm[df_lastfm['artist'] == artist].index[0]
447                       for artist in lastfm_artists}
448
449     artist_to_titles = {}
450     for idx, row in df_lastfm.iterrows():
451         artist = row['artist']
452         title = row['song_name']
453         if artist not in artist_to_titles:
454             artist_to_titles[artist] = set()
455             artist_to_titles[artist].add(title)
456
457     lastfm_artist_title_to_index = {}
458     for idx, row in df_lastfm.iterrows():
459         artist = row['artist']
460         title = row['song_name']
461         lastfm_artist_title_to_index[(artist, title)] = idx
462
463     # Pre-compute word sets
464     lastfm_artist_words_dict = {idx: set(row['artist'].split()) for idx, row in
df_lastfm.iterrows()}
465     lastfm_title_words_dict = {idx: set(row['song_name'].split()) for idx, row in
df_lastfm.iterrows()}
466
467     # Pre-compute Spotify artist and title word sets
468     spot_artist_words_dict = {idx: set(row['top_artist'].split()) for idx, row in
df_spot.iterrows()}
469     spot_title_words_dict = {idx: set(row['song_name'].split()) for idx, row in
df_spot.iterrows()}
470
471     # Create artist+first letter of title index
472     artist_title_first = {}
473     for idx, row in df_lastfm.iterrows():
474         if row['song_name']:
475             key = (row['artist'], row['song_name'][0])
476             if key not in artist_title_first:
477                 artist_title_first[key] = []
478                 artist_title_first[key].append(idx)
479
480     # Process each Spotify track

```

```

481     for index, row in df_spot.iterrows():
482         # Skip if already has a match
483         if df_spot.at[index, 'has_lastfm_match']:
484             continue
485
486         artist = row['top_artist']
487         song_name = row['song_name']
488         match_found = False
489
490         # Get pre-computed word sets for this Spotify track
491         spot_artist_words = spot_artist_words_dict[index]
492         spot_title_words = spot_title_words_dict[index]
493
494         best_match_performer = None
495         best_match_title = None
496         best_match_score = 0
497         best_match_artist_match_pct = 0
498         best_match_name_match_pct = 0
499         best_match_lastfm_index = None
500
501         if index % 10 == 0:
502             print(f"Processing record {index + 1} of {len(df_spot)}...")
503
504         if artist in lastfm_artists:
505             curr_artist = artist
506
507             # Check for exact artist+title match first
508             if song_name in artist_to_titles.get(curr_artist, set()):
509                 lastfm_index = lastfm_artist_title_to_index.get((curr_artist, song_name))
510                 best_match_lastfm_index = lastfm_index
511
512                 # Update both dataframes
513                 df_spot.at[index, 'has_lastfm_match'] = True
514                 df_spot.at[index, 'lastfm_match_index'] = lastfm_index
515                 df_lastfm.at[lastfm_index, 'has_spot_match'] = True
516                 df_lastfm.at[lastfm_index, 'spot_match_index'] = index
517                 df_lastfm.at[lastfm_index, 'exact_match'] = True
518
519                 match_found = True
520                 append_match_result(results, artist, song_name, curr_artist, song_name,
521                                     match_found, best_match_lastfm_index, index, 100, 100,
522                                     100)
523                 continue
524
525             start_index = artist_indices[curr_artist]
526
527             for i in range(start_index, len(df_lastfm)):
528                 if df_lastfm.at[i, 'has_spot_match']:
529                     continue

```

```
530
531     curr_artist = df_lastfm.iloc[i]['artist']
532     title = df_lastfm.iloc[i]['song_name']
533
534     if curr_artist[0] != artist[0]:
535         continue
536     if title[0] != song_name[0]:
537         continue
538
539     curr_artist_words = lastfm_artist_words_dict[i]
540     if not spot_artist_words.intersection(curr_artist_words):
541         continue
542
543     curr_title_words = lastfm_title_words_dict[i]
544     if not spot_title_words.intersection(curr_title_words):
545         continue
546
547     artist_score = fuzz.QRatio(artist, curr_artist)
548     if artist_score < 50:
549         continue
550     title_score = fuzz.QRatio(song_name, title)
551     if title_score < 50:
552         continue
553     geo_mean_score = sqrt(artist_score * title_score)
554
555     if geo_mean_score > best_match_score:
556         best_match_performer = curr_artist
557         best_match_title = title
558         best_match_score = geo_mean_score
559         best_match_artist_match_pct = artist_score
560         best_match_name_match_pct = title_score
561         best_match_lastfm_index = i
562
563     # Need slightly higher threshold for lastfm
564     # Has a few cover bands that foul things up
565     if geo_mean_score >= 75:
566         match_found = True
567
568         lastfm_index = i
569         df_spot.at[index, 'has_lastfm_match'] = True
570         df_spot.at[index, 'lastfm_match_index'] = lastfm_index
571         df_lastfm.at[lastfm_index, 'has_spot_match'] = True
572         df_lastfm.at[lastfm_index, 'spot_match_index'] = index
573
574         append_match_result(results, artist, song_name, curr_artist, title,
575                             match_found, best_match_lastfm_index, index,
576                             best_match_score,
577                             best_match_artist_match_pct, best_match_name_match_pct)
578
579     break
```

```

578
579     if not match_found:
580         for i in range(0, start_index):
581             if df_lastfm.at[i, 'has_spot_match']:
582                 continue
583
584             curr_artist = df_lastfm.iloc[i]['artist']
585             title = df_lastfm.iloc[i]['song_name']
586
587             if curr_artist[0] != artist[0]:
588                 continue
589             if title[0] != song_name[0]:
590                 continue
591
592             curr_artist_words = lastfm_artist_words_dict[i]
593             if not spot_artist_words.intersection(curr_artist_words):
594                 continue
595
596             curr_title_words = lastfm_title_words_dict[i]
597             if not spot_title_words.intersection(curr_title_words):
598                 continue
599
600             artist_score = fuzz.QRatio(artist, curr_artist)
601             if artist_score < 50:
602                 continue # Skip early
603
604             # Only calculate title score if artist score is promising
605             title_score = fuzz.QRatio(song_name, title)
606             if title_score < 50:
607                 continue # Skip early
608
609             geo_mean_score = sqrt(artist_score * title_score)
610
611             if geo_mean_score > best_match_score:
612                 best_match_performer = curr_artist
613                 best_match_title = title
614                 best_match_score = geo_mean_score
615                 best_match_artist_match_pct = artist_score
616                 best_match_name_match_pct = title_score
617                 best_match_lastfm_index = i
618
619             if geo_mean_score >= 75:
620                 match_found = True
621                 lastfm_index = i
622                 df_spot.at[index, 'has_lastfm_match'] = True
623                 df_spot.at[index, 'lastfm_match_index'] = lastfm_index
624                 df_lastfm.at[lastfm_index, 'has_spot_match'] = True
625                 df_lastfm.at[lastfm_index, 'spot_match_index'] = index
626
627             append_match_result(results, artist, song_name, curr_artist, title,

```

```

628     best_match_score,
629     best_match_name_match_pct)
630     break
631
632 else:
633     for i in range(0, len(df_lastfm)):
634         if df_lastfm.at[i, 'has_spot_match']:
635             continue
636
637         curr_artist = df_lastfm.iloc[i]['artist']
638         title = df_lastfm.iloc[i]['song_name']
639
640         if curr_artist[0] != artist[0]:
641             continue
642         if title[0] != song_name[0]:
643             continue
644
645         curr_artist_words = lastfm_artist_words_dict[i]
646         if not spot_artist_words.intersection(curr_artist_words):
647             continue
648
649         curr_title_words = lastfm_title_words_dict[i]
650         if not spot_title_words.intersection(curr_title_words):
651             continue
652
653         artist_score = fuzz.QRatio(artist, curr_artist)
654         if artist_score < 50:
655             continue
656         title_score = fuzz.QRatio(song_name, title)
657         if title_score < 50:
658             continue
659         geo_mean_score = sqrt(artist_score * title_score)
660
661         if geo_mean_score > best_match_score:
662             best_match_performer = curr_artist
663             best_match_title = title
664             best_match_score = geo_mean_score
665             best_match_artist_match_pct = artist_score
666             best_match_name_match_pct = title_score
667             best_match_lastfm_index = i
668
669         if geo_mean_score >= 75:
670             match_found = True
671             lastfm_index = i
672             df_spot.at[index, 'has_lastfm_match'] = True
673             df_spot.at[index, 'lastfm_match_index'] = lastfm_index
674             df_lastfm.at[lastfm_index, 'has_spot_match'] = True
675             df_lastfm.at[lastfm_index, 'spot_match_index'] = index

```

```

676             append_match_result(results, artist, song_name, curr_artist, title,
677                                 match_found, best_match_lastfm_index, index,
678                                 best_match_score,
679                                 best_match_artist_match_pct, best_match_name_match-
680                                 h_pct)
680             break
681
682         if not match_found:
683             append_match_result(results, artist, song_name, best_match_performer,
684                                 best_match_title,
685                                 match_found, best_match_lastfm_index, index,
686                                 best_match_score,
687                                 best_match_artist_match_pct, best_match_name_match_pct)
688
689         # After all matches are found,
690         # Overwrite group6_id in lastfm set based on findings, but preserve existing matches
691         for index, row in df_spot.iterrows():
692             if df_spot.at[index, 'has_lastfm_match']:
693                 lastfm_index = df_spot.at[index, 'lastfm_match_index']
694                 # Only update if the group6_ids are different
695                 if df_lastfm.at[lastfm_index, 'group6_id'] != row['group6_id']:
696                     df_lastfm.at[lastfm_index, 'group6_id'] = df_spot.at[index, 'group6_id']
697
698         end_time = time.time()
699         elapsed_time = end_time - start_time
700         print(f"\nTotal time taken for Spotify and LastFM matching: {elapsed_time:.2f} seconds")
701
702         results_df = pd.DataFrame(results)
703         results_df.sort_values(by=['best_match_score'], ascending=False, inplace=True)
704         results_df.reset_index(drop=True, inplace=True)
705
706         timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
707         match_results_file_name = f"spottolastfm_match_results_{timestamp}.csv"
708         data_dir = os.path.join(os.getcwd(), 'data')
709         os.makedirs(data_dir, exist_ok=True)
710         match_results_file_path = os.path.join(data_dir, match_results_file_name)
711         results_df.to_csv(match_results_file_path, index=False)
712
713         print(f"Successfully wrote to {match_results_file_name}\n")
714
715     # LastFM is already tied to AcousticBrainz via mbid
716     # So this just updates group6_id in AcousticBrainz
717     def lastfm_to_ab(df_lastfm, df_ab):
718         start_time = time.time()
719
720         # Create a dictionary for faster lookup
721         lastfm_mbid_to_group6id = {
722             row['mbid']: row['group6_id']

```

```

723     for idx, row in df_lastfm.iterrows():
724         if pd.notna(row['mbid']) and pd.notna(row['group6_id']):
725     }
726
727     # Track how many were matched
728     match_count = 0
729
730     # Update df_ab rows
731     for idx, row in df_ab.iterrows():
732         if pd.notna(row['mbid']) and row['mbid'] in lastfm_mbid_to_group6id:
733             df_ab.at[idx, 'group6_id'] = lastfm_mbid_to_group6id[row['mbid']]
734             match_count += 1
735
736     end_time = time.time()
737     elapsed_time = end_time - start_time
738     print(f"\nMatched {match_count} AcousticBrainz records with LastFM records by mbid")
739     print(f"Total time taken: {elapsed_time:.2f} seconds")
740
741     return df_ab
742
743 def update_db_after_matching(df_kag, df_lastfm, df_ab):
744     pg_hook = PostgresHook(postgres_conn_id='pg_group6')
745     conn = pg_hook.get_conn()
746
747     with conn.cursor() as cur:
748         # Update Billboard records based on matches found
749         kag_updates = 0
750         for idx, row in df_kag.iterrows():
751             # Only update records that were explicitly matched in the matching process
752             if 'has_spot_match' in df_kag.columns and row['has_spot_match']:
753                 cur.execute("""
754                     UPDATE billboard_chart_data
755                     SET group6_id = %s
756                     WHERE song_name = %s AND top_artist = %s
757                     """, (
758                         row['group6_id'],
759                         row['song_name'],
760                         row['top_artist']
761                     ))
762                 kag_updates += cur.rowcount
763
764             print(f"Updated {kag_updates} rows in billboard_chart_data")
765
766         # Update LastFM tracks
767         lastfm_updates = 0
768         with conn.cursor() as cur:
769             for idx, row in df_lastfm.iterrows():
770                 # Only update records that were explicitly matched
771                 if 'has_spot_match' in df_lastfm.columns and row['has_spot_match']:
772                     # Use mbid as the primary key for updates - LastFM tracks always have an mbid

```

```

773         cur.execute(
774             "UPDATE lastfm_tracks SET group6_id = %s WHERE mbid = %s",
775             (row['group6_id'], row['mbid']))
776         )
777         lastfm_updates += cur.rowcount
778
779     print(f"Updated {lastfm_updates} rows in lastfm_tracks")
780
781     # Update AcousticBrainz
782     ab_updates = 0
783     with conn.cursor() as cur:
784         for idx, row in df_ab.iterrows():
785             # Only update records with valid mbid
786             if pd.notna(row['mbid']):
787                 cur.execute(
788                     "UPDATE acousticbrainz_features SET group6_id = %s WHERE mbid = %s",
789                     (row['group6_id'], row['mbid']))
790             )
791             ab_updates += cur.rowcount
792
793     print(f"Updated {ab_updates} rows in acousticbrainz_features")
794
795     conn.commit()
796     print("All database updates committed")
797
798
799
800 def match_all_tracks():
801     df_spot, df_kag, df_lastfm, df_ab = load_all_from_db()
802
803     df_spot, df_kag = match_spot_to_kag(df_spot, df_kag)
804
805     df_spot, df_lastfm = match_spot_to_lastfm(df_spot, df_lastfm)
806
807     df_ab = lastfm_to_ab(df_lastfm, df_ab)
808
809     update_db_after_matching(df_kag, df_lastfm, df_ab)
810
811
812
813
814 with DAG(
815     'matching_dag',
816     default_args=default_args,
817     description='Dag for fuzzy matching of spotify, kaggle, lastfm, and acousticbrainz
818     data',
819     schedule_interval=None,
820     catchup=False
821     ) as matching_dag:

```

```
822     start_task = EmptyOperator(task_id='start')
823
824     match_all_tracks = PythonOperator(
825         task_id='match_all_tracks',
826         python_callable=match_all_tracks
827     )
828
829     end_task = EmptyOperator(task_id='end')
830
831     start_task >> match_all_tracks >> end_task
832
833
```