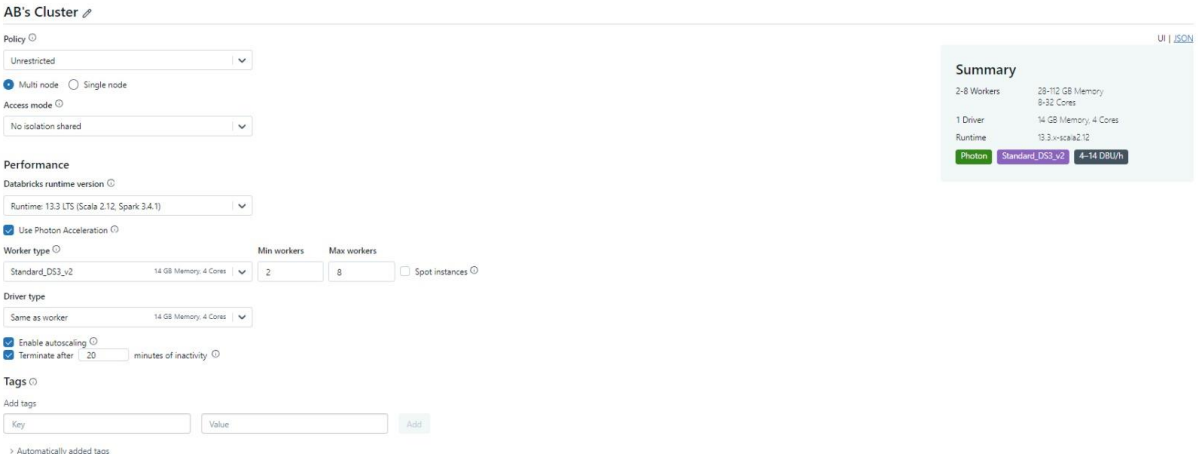
Saransh Rawat (2022OG04027)
Ayush Bansal (2022OG04042)

Use case: **Spotify Recommendation System Platform**

Task 3: **OLAP Queries**

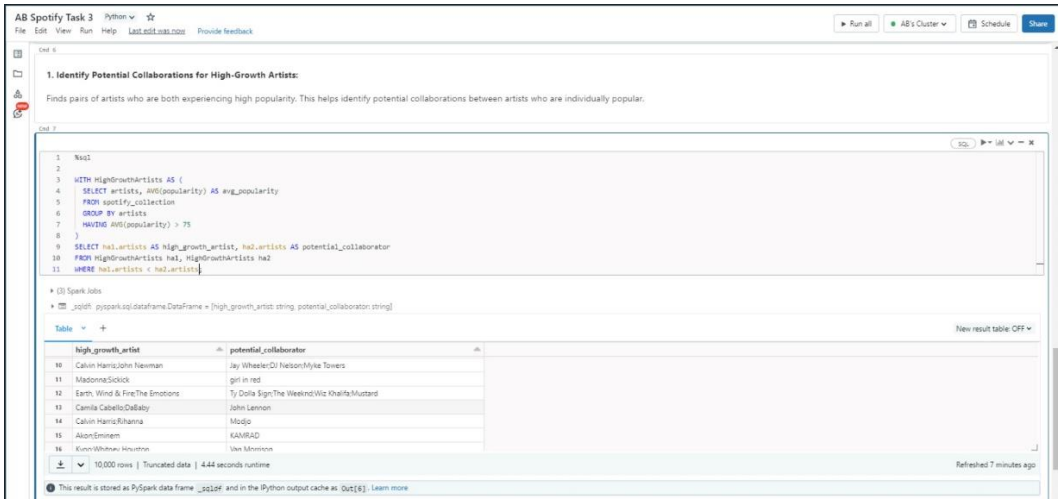
Spark Setup: For this we have use Data Bricks for Spark Infra setup below are the config details



Post completion of the Spark setup here are the few OLAP queries with respect to the Spotify Solution which helps to improve growth/sales

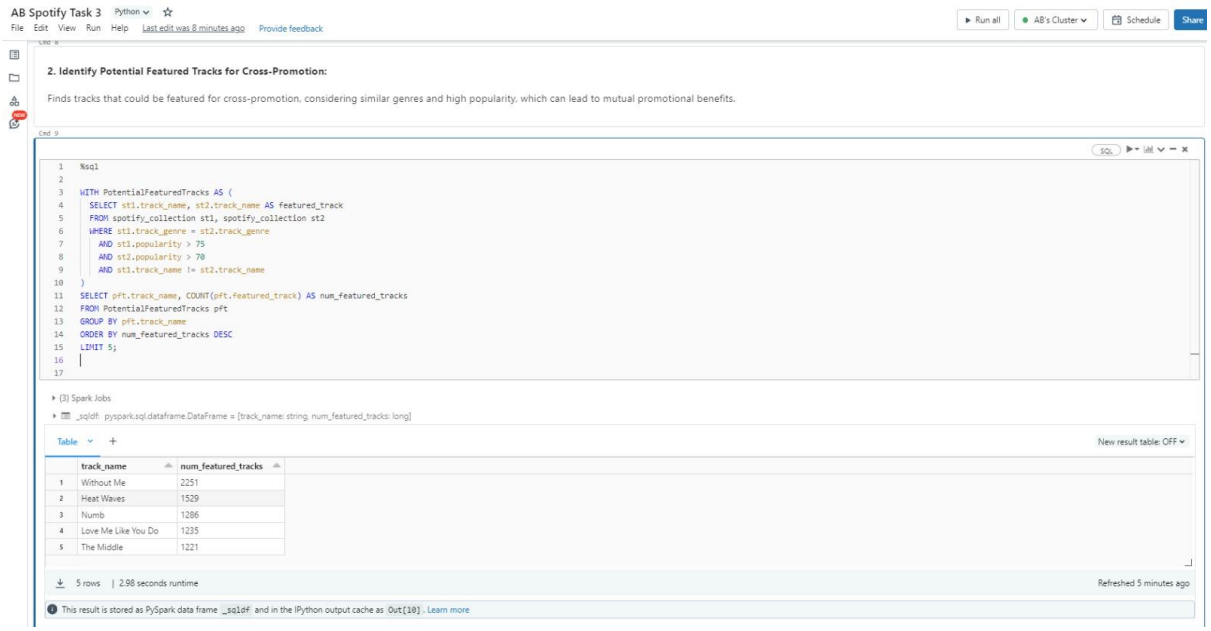
1. **Identify Potential Collaborations for High-Growth Artists:** Finds pairs of artists who are both experiencing high popularity. This helps identify potential collaborations between artists who are individually popular.

```
WITH HighGrowthArtists AS (
  SELECT artists, AVG(popularity) AS avg_popularity
  FROM spotify_collection
  GROUP BY artists
  HAVING AVG(popularity) > 75
)
SELECT ha1.artists AS high_growth_artist, ha2.artists AS potential_collaborator
FROM HighGrowthArtists ha1, HighGrowthArtists ha2
WHERE ha1.artists < ha2.artists;
```



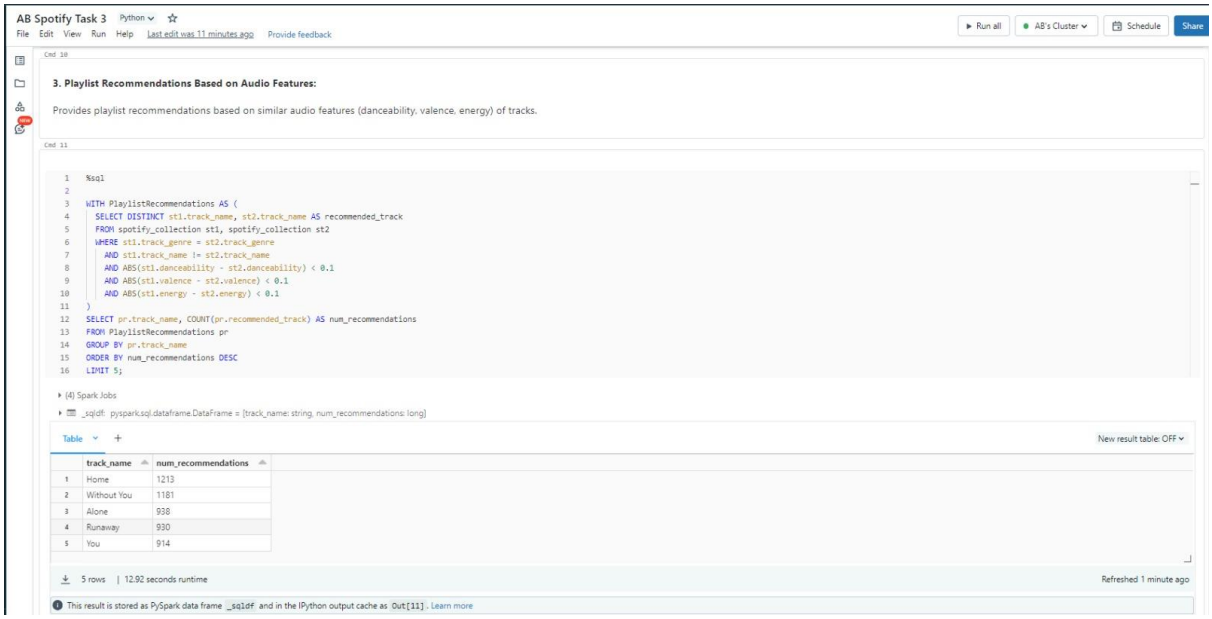
2. **Identify Potential Featured Tracks for Cross-Promotion:** Finds tracks that could be featured for cross-promotion, considering similar genres and high popularity, which can lead to mutual promotional benefits.

```
WITH PotentialFeaturedTracks AS (
 SELECT st1.track_name, st2.track_name AS featured_track
 FROM spotify_collection st1, spotify_collection st2
 WHERE st1.track_genre = st2.track_genre
  AND st1.popularity > 75
  AND st2.popularity > 70
  AND st1.track_name != st2.track_name
)
SELECT pft.track_name, COUNT(pft.featured_track) AS num_featured_tracks
FROM PotentialFeaturedTracks pft
GROUP BY pft.track_name
ORDER BY num_featured_tracks DESC
LIMIT 5;
```



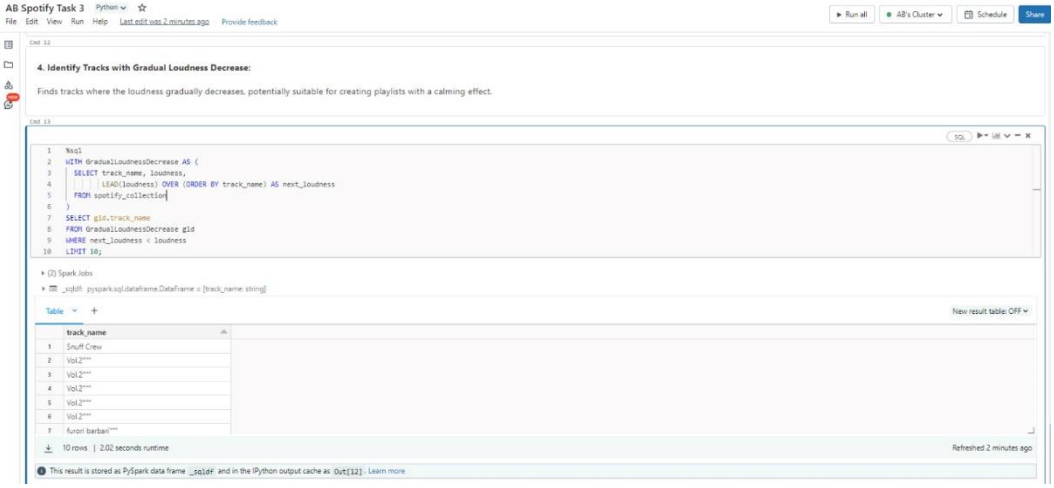
3. **Playlist Recommendations Based on Audio Features:** Provides playlist recommendations based on similar audio features (danceability, valence, energy) of tracks.

```
WITH PlaylistRecommendations AS (
 SELECT DISTINCT st1.track_name, st2.track_name AS recommended_track
 FROM spotify_collection st1, spotify_collection st2
 WHERE st1.track_genre = st2.track_genre
  AND st1.track_name != st2.track_name
  AND ABS(st1.danceability - st2.danceability) < 0.1
  AND ABS(st1.valence - st2.valence) < 0.1
  AND ABS(st1.energy - st2.energy) < 0.1
)
SELECT pr.track_name, COUNT(pr.recommended_track) AS num_recommendations
FROM PlaylistRecommendations pr
GROUP BY pr.track_name
ORDER BY num_recommendations DESC
LIMIT 5;
```



4. **Identify Tracks with Gradual Loudness Decrease:** Finds tracks where the loudness gradually decreases, potentially suitable for creating playlists with a calming effect.

```
WITH GradualLoudnessDecrease AS (
 SELECT track_name, loudness,
      LEAD(loudness) OVER (ORDER BY track_name) AS next_loudness
 FROM spotify_table
)
SELECT gld.track_name
FROM GradualLoudnessDecrease gld
WHERE next_loudness < loudness
LIMIT 10;
```


AB Spotify Task 3
Python
File Edit View Run Help Last edit was 2 minutes ago Provide feedback
Run all
AB's Cluster
Schedule
Share
Cmd 12
4. Identify Tracks with Gradual Loudness Decrease:
Finds tracks where the loudness gradually decreases, potentially suitable for creating playlists with a calming effect.
Cmd 13
SQL
%sql
WITH GradualLoudnessDecrease AS (
SELECT track_name, loudness,
LEAD(loudness) OVER (ORDER BY track_name) AS next_loudness
FROM spotify_collection
)
SELECT gld.track_name
FROM GradualLoudnessDecrease gld
WHERE next_loudness < loudness
LIMIT 10;
(2) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [track_name: string]
Table
New result table: OFF
track_name
1 Snuff Crew
2 Vol.2****
3 Vol.2****
4 Vol.2****
5 Vol.2****
6 Vol.2****
7 furori barbari****
10 rows | 2.02 seconds runtime
Refreshed 2 minutes ago
This result is stored as PySpark data frame _sqldf and in the IPython output cache as Out[12]. Learn more