

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

AB's Cluster

Policy

Unrestricted

☒ Multi node ☐ Single node

Access mode

No isolation shared

Performance

Databricks runtime version

Runtime: 13.3 LTS (Scala 2.12, Spark 3.4.1)

☒ Use Photon Acceleration

Worker type

Standard_DS3_v2 14 GB Memory, 4 Cores

Min workers: 2 Max workers: 8

☐ Spot instances

Driver type

Same as worker 14 GB Memory, 4 Cores

☒ Enable autoscaling

☒ Terminate after 20 minutes of inactivity

Tags

Add tags

Key Value Add

> Automatically added tags

Summary

2-8 Workers 28-112 GB Memory 8-32 Cores

1 Driver 14 GB Memory, 4 Cores

Runtime 13.3.x-scala2.12

Photon Standard_DS3_v2 4-14 DBU/h

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;
```

AB Spotify Task 3 Python ☆

File Edit View Run Help Last edit was 8 minutes ago Provide feedback

Run all AB's Cluster Schedule Share

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.

```

1  --sql
2
3  WITH HighGrowthArtists AS (
4    SELECT artists, avg(popularity) AS avg_popularity
5    FROM spotify_collection
6    GROUP BY artists
7    HAVING avg(popularity) > 75
8  )
9  SELECT hga1.artists AS high_growth_artist, hga2.artists AS potential_collaborator
10 FROM HighGrowthArtists hga1, HighGrowthArtists hga2
11 WHERE hga1.artists < hga2.artists

```

(3) Spark Jobs

__sql__: pyspark.sql.dataframe.DataFrame = [high_growth_artist string, potential_collaborator string]

high_growth_artist	potential_collaborator
Calvin Harris/John Newman	Jay Wheeler/DJ Nelson/Myke Towers
Madonna/Sickick	girl in red
Earth, Wind & Fire/The Emotions	Ty Dolla Sign/The Weeknd/Wiz Khalifa/Mustard
Camila Cabello/Dababy	John Lennon
Calvin Harris/Diana	Moby
Alison/Eminem	Kanye West
Kanye West/Dr. Dre	Dr. Dre

10,000 rows | Truncated data | 4.44 seconds runtime

Refreshed 7 minutes ago

This result is stored as PySpark data frame __sql__ and in the Python output cache as: Out[6]. Learn more

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;

```

AB Spotify Task 3 Python ☆

File Edit View Run Help Last edit was 8 minutes ago Provide feedback

Run all AB's Cluster Schedule Share

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.

```

1  --sql
2
3  WITH PotentialFeaturedTracks AS (
4    SELECT st1.track_name, st2.track_name AS featured_track
5    FROM spotify_collection st1, spotify_collection st2
6    WHERE st1.track_genre = st2.track_genre
7    AND st1.popularity > 75
8    AND st2.popularity > 70
9    AND st1.track_name != st2.track_name
10 )
11 SELECT pft.track_name, COUNT(pft.featured_track) AS num_featured_tracks
12 FROM PotentialFeaturedTracks pft
13 GROUP BY pft.track_name
14 ORDER BY num_featured_tracks DESC
15 LIMIT 5;
16
17

```

(3) Spark Jobs

__sql__: pyspark.sql.dataframe.DataFrame = [track_name: string, num_featured_tracks: long]

track_name	num_featured_tracks
Without Me	2251
Heat Waves	1529
Numb	1286
Love Me Like You Do	1235
The Middle	1221

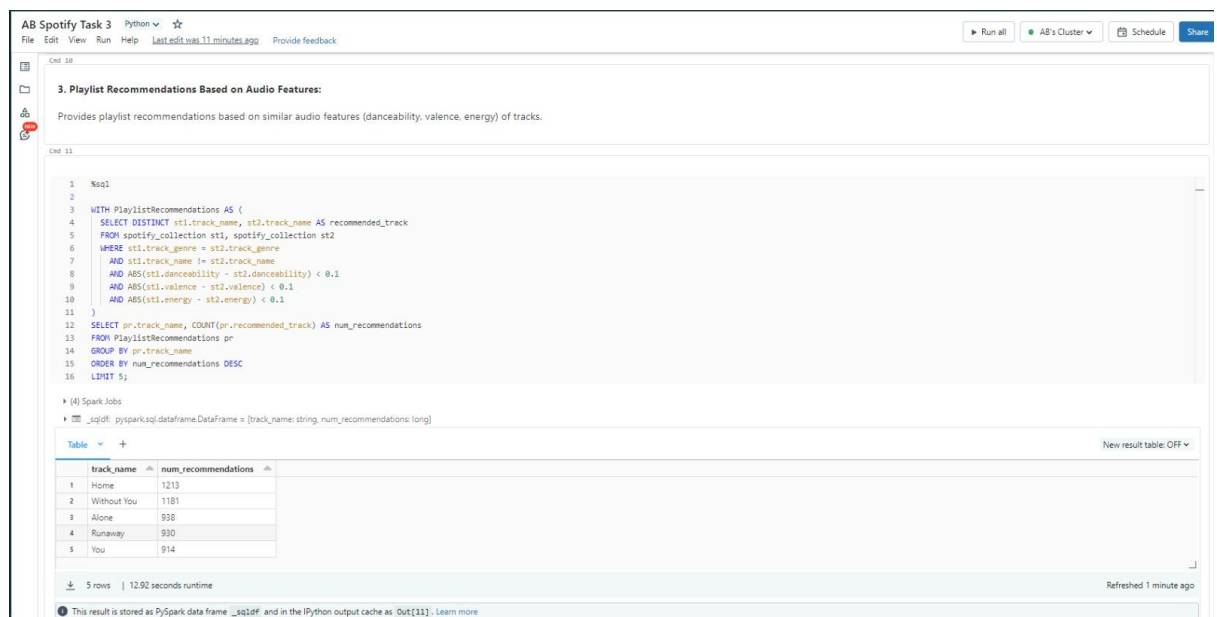
5 rows | 2.98 seconds runtime

Refreshed 5 minutes ago

This result is stored as PySpark data frame __sql__ and in the Python output cache as: Out[10]. Learn more

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;
```



The screenshot shows a Databricks workspace interface. At the top, there's a header for 'AB Spotify Task 3' with a Python icon and a star. Below the header, there's a toolbar with 'Run all', 'AB's Cluster', 'Schedule', and 'Share' buttons. The main area displays a SQL query titled '3. Playlist Recommendations Based on Audio Features:'. The query is as follows:

```
1 %sql  
2  
3 WITH PlaylistRecommendations AS (  
4   SELECT DISTINCT st1.track_name, st2.track_name AS recommended_track  
5   FROM spotify_collection st1, spotify_collection st2  
6   WHERE st1.track_genre = st2.track_genre  
7     AND st1.track_name != st2.track_name  
8     AND ABS(st1.danceability - st2.danceability) < 0.1  
9     AND ABS(st1.valence - st2.valence) < 0.1  
10    AND ABS(st1.energy - st2.energy) < 0.1  
11  )  
12 SELECT pr.track_name, COUNT(pr.recommended_track) AS num_recommendations  
13 FROM PlaylistRecommendations pr  
14 GROUP BY pr.track_name  
15 ORDER BY num_recommendations DESC  
16 LIMIT 5;
```

Below the query, there's a section for '(4) Spark Jobs' showing a job named '_sqldf: pyspark.sql.dataframe.DataFrame = [track_name: string, num_recommendations: long]'. Below this, there's a table view showing the results of the query:

track_name	num_recommendations
Home	1213
Without You	1181
Alone	938
Runaway	930
You	914

At the bottom, there's a status bar indicating '5 rows | 12.92 seconds runtime' and 'Refreshed 1 minute ago'. A note at the very bottom states: 'This result is stored as PySpark data frame '_sqldf' and in the IPython output cache as 'Out[11]'. Learn more'.

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 3Python☆

FileEditViewRunHelpLast edit was 2 minutes agoProvide feedback

Run allAB's ClusterScheduleShow

Out 11

4. Identify Tracks with Gradual Loudness Decrease:

Finds tracks where the loudness gradually decreases, potentially suitable for creating playlists with a calming effect.

Out 12

```
1  %sql
2  %%sql <gradualLoudnessDecrease AS (
3  |  SELECT track_name, loudness,
4  |  |  LEAD(loudness) OVER (ORDER BY track_name) AS next_loudness
5  |  FROM spotify_collection)
6  |
7  SELECT g1.track_name
8  FROM gradualLoudnessDecrease g1d
9  WHERE next_loudness < loudness
10 LIMIT 10;
```

Spark Jobs

__sql__ pySpark.sql.DataFrame.DataFrame = [track_name:string]

Table

track_name	
1	Snuff Crew
2	Vol 2****
3	Vol 2****
4	Vol 2****
5	Vol 2****
6	Vol 2****
7	Aurori barkan****

10 rows | 2.02 seconds runtime

Refreshed 2 minutes ago

This result is stored as PySpark data frame __sql__ and in the IPython output cache as Out[12]. Learn more