databricks Top Pop-Punk & Metalcore Songs - Spotify

(https://databricks.com)

1: **Import Libraries**

```
import time
import requests
import pandas as pd
from pyspark.sql import SparkSession
```

2: **Spotify API Credentials**

```
CLIENT_ID = "4e85871bfd71487db76f309451192a58"
CLIENT_SECRET = "5d2bd0dac36647ab8318a20e7710c5cb"
```

3: **Get Spotify Token**

```
# Get Access Token
def get_spotify_token(client_id, client_secret):
    url = "https://accounts.spotify.com/api/token"
    headers = {"Content-Type": "application/x-www-form-urlencoded"}
    data = {"grant_type": "client_credentials"}
    response = requests.post(
        url, headers=headers, data=data, auth=(client_id, client_secret)
    )
    response.raise_for_status()
    return response.json().get("access_token")
# Print Token
print(get_spotify_token(client_id=CLIENT_ID,client_secret=CLIENT_SECRET))
```

```
BQCDNfI_m8oixafToM-PKZZLNXBzuVX0tALfSmOhFYA8zQc704ENRNJmcHMo0iwk0z-t_nh-w_T8zaXvXl-Eb7wHSaSyoKCKm3AuHhHjfcMjxkLwYLk
```

4: **GET top tracks with retry logic**

```
def fetch_top_tracks(artist_id, token, market="US", limit=20):
    url = f"https://api.spotify.com/v1/artists/{artist_id}/top-tracks"
    headers = {"Authorization": f"Bearer {token}"}
    params = {"market": market}

    for attempt in range(5):  # Retry 5x
        response = requests.get(url, headers=headers, params=params)

        if response.status_code == 200:  # 200: Success, 403 invalid tokens, 404: error in call
            tracks = response.json().get("tracks", [])[:limit]
            return [
                {
                    "Artist_ID": artist_id,
                    "Track_ID": track.get("id"),
                    "Track_Name": track.get("name"),
                    "Popularity": track.get("popularity", 0),
                    "Duration_ms": track.get("duration_ms"),
                    "Explicit": track.get("explicit"),
                    "Album_Name": track.get("album", {}).get("name", "Unknown"),
                    "Album_Release_Date": track.get("album", {}).get("release_date", "Unknown"),
                    "Release_Date_Precision": track.get("album", {}).get("release_date_precision", "Unknown"),
                    "Track_Number": track.get("track_number", 0),
                    "Disc_Number": track.get("disc_number", 0),
                    "Available_Markets": len(track.get("available_markets", [])),
                    "External_URL": track.get("external_urls", {}).get("spotify", "Unknown"),
                    "Artists_Featured": ", ".join([artist["name"] for artist in track.get("artists", [])]),
                    "Album_Type": track.get("album", {}).get("album_type", "Unknown"),
                    "Is_Local": track.get("is_local", False),
                    "Href": track.get("href", "Unknown"),
                }
                for track in tracks
            ]

        elif response.status_code == 429:  # Rate limit exceeded
            retry_after = int(response.headers.get("Retry-After", 1))
            print(f"Rate limit exceeded. Retrying after {retry_after} seconds...")
            time.sleep(retry_after)

        else:
            response.raise_for_status()

    print(f"Failed to fetch top tracks for artist {artist_id} after multiple retries.")
    return []
```

5: **Initialize Spark**

```
# Initiate Spark Session
spark = SparkSession.builder.appName("SpotifyTrackFetcher").enableHiveSupport().getOrCreate()
# Validate session
print(spark.catalog)

# GET Spotify token
token = get_spotify_token(CLIENT_ID, CLIENT_SECRET)

# Load  artist df
artist_spark_df = spark.sql("SELECT * FROM artists_table2")

# Convert Spark DF to Pandas DF
artist_df = artist_spark_df.toPandas()
```

▸ artist_spark_df: pyspark.sql.dataframe.DataFrame = [Artist_Name: string, Artist_ID: string ... 4 more fields]

```
<pyspark.sql.catalog.Catalog object at 0x7f4b89193cd0>
```

6: **GET top tracks with retry and throttling**

```
all_track_data = []
for _, row in artist_df.iterrows():
    artist_id = row["Artist_ID"]
    top_tracks = fetch_top_tracks(artist_id, token)
    time.sleep(0.5)  # Delay to for rate limits
    all_track_data.extend(top_tracks)

tracks_df = pd.DataFrame(all_track_data)
print(tracks_df.head())
```

```
   Track_Number  Disc_Number  Available_Markets  \
0            11            1                  0
1             5            1                  0
2             1            1                  0
3             6            1                  0
4             5            1                  0

                                      External_URL     Artists_Featured  \
0  https://open.spotify.com/track/7j31rVgGX9Q2blT (https://open.spotify.com/track/7j31rVgGX9Q2blT)...  My Chemica
l Romance
1  https://open.spotify.com/track/5wQnmLuC1W7ATsA (https://open.spotify.com/track/5wQnmLuC1W7ATsA)...  My Chemica
l Romance
2  https://open.spotify.com/track/5dTHtzHFPyi8TlT (https://open.spotify.com/track/5dTHtzHFPyi8TlT)...  My Chemica
l Romance
3  https://open.spotify.com/track/4RAOI1etsgbh5NP (https://open.spotify.com/track/4RAOI1etsgbh5NP)...  My Chemica
l Romance
4  https://open.spotify.com/track/7lRlq939cDG4SzW (https://open.spotify.com/track/7lRlq939cDG4SzW)...  My Chemica
l Romance

  Album_Type  Is_Local                                              Href
```

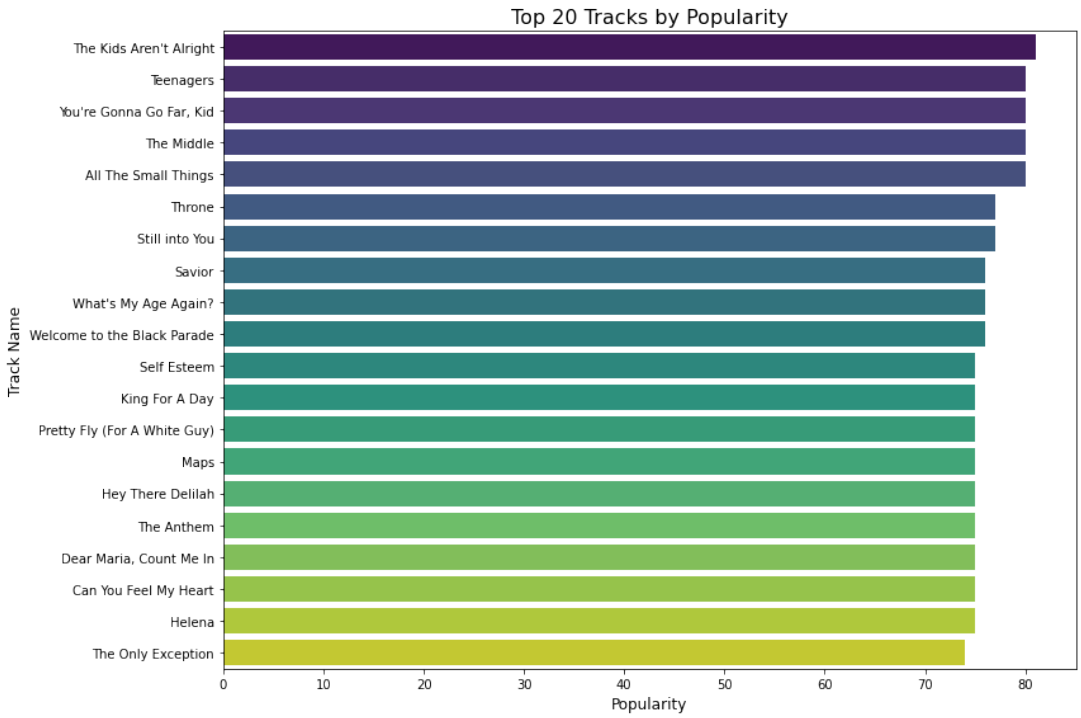
7

```
import seaborn as sns
import matplotlib.pyplot as plt

# Filter data (Top 20 tracks by Popularity)
top_tracks = tracks_df.sort_values(by='Popularity', ascending=False).head(20)

# Create a barchart
plt.figure(figsize=(12, 8))
sns.barplot(
    x='Popularity',
    y='Track_Name',
    data=top_tracks,
    palette='viridis'
)

# Set plot title & axis labels
plt.title('Top 20 Tracks by Popularity', fontsize=16)
plt.xlabel('Popularity', fontsize=12)
plt.ylabel('Track Name', fontsize=12)
plt.tight_layout()
plt.show()
```



8: **Convert to Spark DF and save**

```
tracks_spark_df = spark.createDataFrame(tracks_df)
tracks_spark_df.write.option("mergeSchema", "true").mode("overwrite").saveAsTable("default.top_tracks_table")

print("Top tracks data saved to Hive table 'default.top_tracks_table'.")
```

▸ tracks_spark_df: pyspark.sql.dataframe.DataFrame = [Artist_ID: string, Track_ID: string ... 15 more fields]

```
Top tracks data saved to Hive table 'default.top_tracks_table'.
```

9: **Top 20 Tracks by Artist**

▸ _sqldf: pyspark.sql.dataframe.DataFrame = [Track_Name: string, Artists_Featured: string ... 4 more fields]

Table