databricks Top Pop-Punk & Metalcore Artists - Spotify

(https://databricks.com)

1: **Import Libraries**

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

2: **Enter Spotify API Credentials**

```
CLIENT_ID = "01610abcafbc4bc6899f1217cd4407a9"
CLIENT_SECRET = "0ad83c8b2e3642b38a7ad2aaa58ab3ed"
```

3: **Input Genres (5 max)**

```
# Genres to process
GENRES = [
    "pop-punk",
    "metalcore"

]
```

4: **Get Spotify 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")
```

5: **GET Top Artists by Genre**

```
def fetch_top_artists_by_genre(genre, token, limit=50):
    url = "https://api.spotify.com/v1/search"
    headers = {"Authorization": f"Bearer {token}"}
    params = {"q": f"genre:{genre}", "type": "artist", "limit": limit}
    response = requests.get(url, headers=headers, params=params)
    response.raise_for_status()
    return response.json().get("artists", {}).get("items", [])
```

6: **GET All Artists by Genres**

```
def fetch_all_artists_by_genres(genres, token, limit=50):
    """
    Fetch artists for multiple genres and append to a global list.

    Args:
    - genres (list): List of genres to process.
    - token (str): Spotify API token.
    - limit (int): Number of artists to fetch per genre (default: 50).

    Returns:
    - List of artist data dictionaries.
    """
    # Data storage
    artist_data = []

    for genre in genres:
        print(f"Fetching artists for genre: {genre}")
        artists = fetch_top_artists_by_genre(genre, token, limit)

        if not artists:
            print(f"No artists found for genre: {genre}")
            continue

        for artist in artists:
            artist_data.append(
                {
                    "Artist_Name": artist.get("name", "Unknown"),  # Cleaned column names
                    "Artist_ID": artist.get("id", "Unknown"),
                    "Genre": genre,
                    "Popularity": artist.get("popularity", 0),
                    "Followers": artist.get("followers", {}).get("total", 0),
                    "Spotify_URL": artist.get("external_urls", {}).get("spotify", ""),
                }
            )
    return artist_data
```

7: **Initialize Spark**

```python
spark = SparkSession.builder \
    .appName("Spotify Artist Data") \
    .enableHiveSupport() \
    .getOrCreate()

# Get spotify token
token = get_spotify_token(CLIENT_ID, CLIENT_SECRET)

# Get artist data
artist_data = fetch_all_artists_by_genres(GENRES, token, limit=50)

# Validate data pulll
if not artist_data:
    print("No artist data fetched. Please check the API or genre list.")
else:
    # Convert to Pandas DF
    artist_df = pd.DataFrame(artist_data)
    print("\nArtists Data (Pandas DataFrame):")
    print(artist_df.head())
    artist_df = artist_df.drop_duplicates(subset=["Artist_Name"], keep="first")

    # Pandas DF to Spark DF
    spark_df = spark.createDataFrame(artist_df)

    # Clean columns in Spark DF
    for col in spark_df.columns:
        spark_df = spark_df.withColumnRenamed(col, col.replace(" ", "_").replace(".", "_"))

    print("\nSpark DataFrame Schema:")
    spark_df.printSchema()

    # Save to Hive table w/ schema merge
    spark_df.write.option("mergeSchema", "true") \
        .mode("overwrite") \
        .saveAsTable("default.artists_table2")

    print("Data saved to Hive metastore with schema merge.")
```

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

```
2     Pierce The Veil  4iJLPqClelZOBCBifm8Fzv  pop-punk         76
3           blink-182  6FBDaR13swtiWwGhX1WQsP  pop-punk         79
4       The Offspring  5LfGQac0EIXyAN8aUwmNAQ  pop-punk         79

   Followers                                        Spotify_URL
0    9237236  https://open.spotify.com/artist/7FBcuc1gsnv6Y1 (https://open.spotify.com/artist/7FBcuc1gsnv6Y1)...
1    8989492  https://open.spotify.com/artist/74XFHRwlV6OrjE (https://open.spotify.com/artist/74XFHRwlV6OrjE)...
2    3321641  https://open.spotify.com/artist/4iJLPqClelZOBC (https://open.spotify.com/artist/4iJLPqClelZOBC)...
3    8700731  https://open.spotify.com/artist/6FBDaR13swtiWw (https://open.spotify.com/artist/6FBDaR13swtiWw)...
4    5999240  https://open.spotify.com/artist/5LfGQac0EIXyAN (https://open.spotify.com/artist/5LfGQac0EIXyAN)...

Spark DataFrame Schema:
root
 |-- Artist_Name: string (nullable = true)
 |-- Artist_ID: string (nullable = true)
 |-- Genre: string (nullable = true)
 |-- Popularity: long (nullable = true)
 |-- Followers: long (nullable = true)
 |-- Spotify_URL: string (nullable = true)

Data saved to Hive metastore with schema merge.
```

8: **Remove duplicate names**

```
# Remove duplicate artist names keeping the first occurrence
if not artist_df.empty:
    artist_df = artist_df.drop_duplicates(subset=["Artist_Name"], keep="first")
    print("\nArtists Data After Removing Duplicates (Pandas DataFrame):")
    print(artist_df.head())
else:
    print("No data available to process.")
```

```

Artists Data After Removing Duplicates (Pandas DataFrame):
           Artist_Name               Artist_ID     Genre  Popularity  \
0  My Chemical Romance  7FBcuc1gsnv6Y1nwFtNRCb  pop-punk          81
1             Paramore  74XFHRwlV6OrjEM0A2NCMF  pop-punk          80
2       Pierce The Veil  4iJLPqClelZOBCBifm8Fzv  pop-punk          76
3            blink-182  6FBDaR13swtiWwGhX1WQsP  pop-punk          79
4        The Offspring  5LfGQac0EIXyAN8aUwmNAQ  pop-punk          79

   Followers                                        Spotify_URL
0    9237236  https://open.spotify.com/artist/7FBcuc1gsnv6Y1 (https://open.spotify.com/artist/7FBcuc1gsnv6Y1)...
1    8989492  https://open.spotify.com/artist/74XFHRwlV6OrjE (https://open.spotify.com/artist/74XFHRwlV6OrjE)...
2    3321641  https://open.spotify.com/artist/4iJLPqClelZOBC (https://open.spotify.com/artist/4iJLPqClelZOBC)...
3    8700731  https://open.spotify.com/artist/6FBDaR13swtiWw (https://open.spotify.com/artist/6FBDaR13swtiWw)...
4    5999240  https://open.spotify.com/artist/5LfGQac0EIXyAN (https://open.spotify.com/artist/5LfGQac0EIXyAN)...
```

9: **Top 20 Artists by Followers**

```
%sql

select
Artist_Name,
Genre,
Popularity,
Followers,
Spotify_URL
from artists_table2
order by
4 desc
limit 20
```

_sqldf: pyspark.sql.dataframe.DataFrame = [Artist_Name: string, Genre: string ... 3 more fields]

Table

10: **Top 20 Artists by Popularity**

```
_sqldf:  pyspark.sql.dataframe.DataFrame = [Artist_Name: string, Genre: string ... 3 more fields]
```

Table

This result is stored as `_sqldf` and can be used in other Python cells.