2024-11-04

DATABASES 2

LAB 6 - POSTGRES, PYTHON AND EXTERNAL APIS

Paulina Czarnota C21365726

INTRODUCTION

This lab demonstrates how to integrate PostgreSQL, Python, and external APIs to process song lyrics, analyse word frequencies, and explore differences between two distinct musical eras (2023 vs. 1982). The objective is to store song metadata and lyrics, process them into structured data, and extract meaningful insights. This exercise showcases automated data collection, data transformation, and query-based analysis, relevant in areas like sentiment analysis, trend prediction, and market research.

PROBLEM DESCRIPTION

The tasks for this lab are as follows:

1. **Database Setup:**
    - Create two tables: songs and songs_words.
    - Populate the songs table with metadata for the top 10 songs from 2023 and 1982.
2. **Python Integration:**
    - Connect to the database and fetch song data into a Pandas DataFrame.
    - Use the Lyrics.ovh API to fetch lyrics and process word frequency data.
    - Populate the songs_words table with processed word data.
3. **Data Cleanup and Analysis:**
    - Remove short words (less than 4 characters) from the songs_words table.
    - Identify the top 5 most-used words for songs from 2023 and 1982.

DATABASE SETUP

The songs.sql script was used to create the database tables and populate the songs table. Below is the table structure:

1. **Table: songs**
    - **Columns:**
        - song_title (VARCHAR(300), PRIMARY KEY)
        - singer (VARCHAR(100))
        - song_year (INT)
2. **Table: songs_words**
    - **Columns:**
        - song_title (VARCHAR(300), FOREIGN KEY REFERENCES songs(song_title))
        - word (VARCHAR(40))

- word_count (INT)

POSTGRESQL SCRIPT TO CREATE TABLES AND INSERT DATA

```
-- Drop the existing tables if needed
DROP TABLE IF EXISTS songs_words;
DROP TABLE IF EXISTS songs;

-- Create the songs table
CREATE TABLE songs (
   song_title VARCHAR(300) PRIMARY KEY,
   singer VARCHAR(100),
   song_year INT
);

-- Create the songs_words table with unique constraints
CREATE TABLE songs_words (
   song_title VARCHAR(300) REFERENCES songs(song_title),
   word VARCHAR(100),
   word_count INT,
   CONSTRAINT unique_song_word UNIQUE (song_title, word)
);

-- Insert songs into the songs table
INSERT INTO songs (song_title, singer, song_year)
VALUES
   ('Last Night', 'Morgan Wallen', 2023),
   ('Flowers', 'Miley Cyrus', 2023),
   ('Kill Bill', 'SZA', 2023),
   ('Anti-Hero', 'Taylor Swift', 2023),
   ('Creepin', 'Metro Boomin', 2023),
   ('Calm Down', 'Rema', 2023),
   ('Die For You', 'The Weeknd', 2023),
   ('Fast Car', 'Luke Combs', 2023),
   ('Snooze', 'SZA', 2023),
   ('Physical', 'Olivia Newton-John', 1982),
   ('Eye of the Tiger', 'Survivor', 1982),
   ('I Love Rock n Roll', 'Joan Jett', 1982),
   ('Ebony and Ivory', 'Paul McCartney', 1982),
   ('Centerfold', 'The J. Geils Band', 1982),
   ('Jack and Diane', 'John Cougar', 1982),
   ('Hurts So Good', 'John Cougar', 1982),
   ('Dont You Want Me', 'Human League', 1982),
   ('Abracadabra', 'Steve Miller Band', 1982),
   ('Hard to Say I''m Sorry', 'Chicago', 1982)
ON CONFLICT (song_title) DO NOTHING;
```

EXPLANATION

- The songs table stores metadata about songs, including title, singer, and year.
- The songs_words table captures word frequency data, linking each word to its song title.

PYTHON PROGRAMS

1. Connecting to the Database and Fetching Songs

The script connect_to_db.py connects to the database, fetches data from the songs table, and displays it as a Pandas DataFrame.

```
import psycopg2
import pandas as pd

# Connect to your PostgreSQL database
try:
    conn = psycopg2.connect(
        dbname="postgres",
        user="postgres",
        password="nowe_haslo",
        host="localhost",
        port="5432"
    )
    # Fetch the data from the songs table
    cur = conn.cursor()
    cur.execute("SELECT * FROM songs;")
    rows = cur.fetchall()
    # Convert the data to a pandas DataFrame
    df = pd.DataFrame(rows, columns=["song_title", "singer", "song_year"])
    print(df)
except Exception as e:
    print(f"Error: {e}")
finally:
    if conn:
        cur.close()
        conn.close()
```

2. Fetching Lyrics and Populating songs_words

The script populate_songs_words.py uses the Lyrics.ovh API to fetch lyrics, processes them into words, and populates the songs_words table.

```
import requests
import psycopg2
from collections import Counter


def get_lyrics(artist, song_title):
    """Fetch lyrics from Lyrics.ovh API."""
    url = f"https://api.lyrics.ovh/v1/{artist}/{song_title}"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json().get("lyrics", "")
    else:
        print(f"Failed to fetch lyrics for {song_title} by {artist}. Status Code:
{response.status_code}")
        return ""

# Connect to PostgreSQL database
try:
    conn = psycopg2.connect(
        dbname="postgres",
        user="postgres",
        password="nowe_haslo",
        host="localhost",
        port="5432"
    )
    cur = conn.cursor()
    # Fetch songs from the database
    cur.execute("SELECT song_title, singer FROM songs;")
    songs = cur.fetchall()

    # Iterate over each song and populate songs_words
    for song_title, singer in songs:
        lyrics = get_lyrics(singer, song_title)
        if lyrics:
            word_counts = Counter(lyrics.split())
            for word, count in word_counts.items():
                if 4 <= len(word) <= 100:  # Ignore words shorter than 4 characters
                    cur.execute("""
                        INSERT INTO songs_words (song_title, word, word_count)
                        VALUES (%s, %s, %s)
                        ON CONFLICT (song_title, word)
                        DO UPDATE SET word_count = songs_words.word_count + EXCLUDED.word_count;
                    """, (song_title, word, count))
    conn.commit()
    print("Lyrics and word counts have been successfully inserted.")
except Exception as e:
    print(f"Error: {e}")
finally:
    if conn:
        cur.close()
        conn.close()
```

DATA CLEANUP AND ANALYSIS

Data Cleanup

1. Remove words shorter than 4 characters:

```
DELETE FROM songs_words WHERE LENGTH(word) < 4;
```

Data Analysis Queries

2. Find the top 5 most-used words for 2023:

```
SELECT word, SUM(word_count) AS total_count
FROM songs_words
JOIN songs ON songs_words.song_title = songs.song_title
WHERE song_year = 2023
GROUP BY word
ORDER BY total_count DESC
LIMIT 5;
```

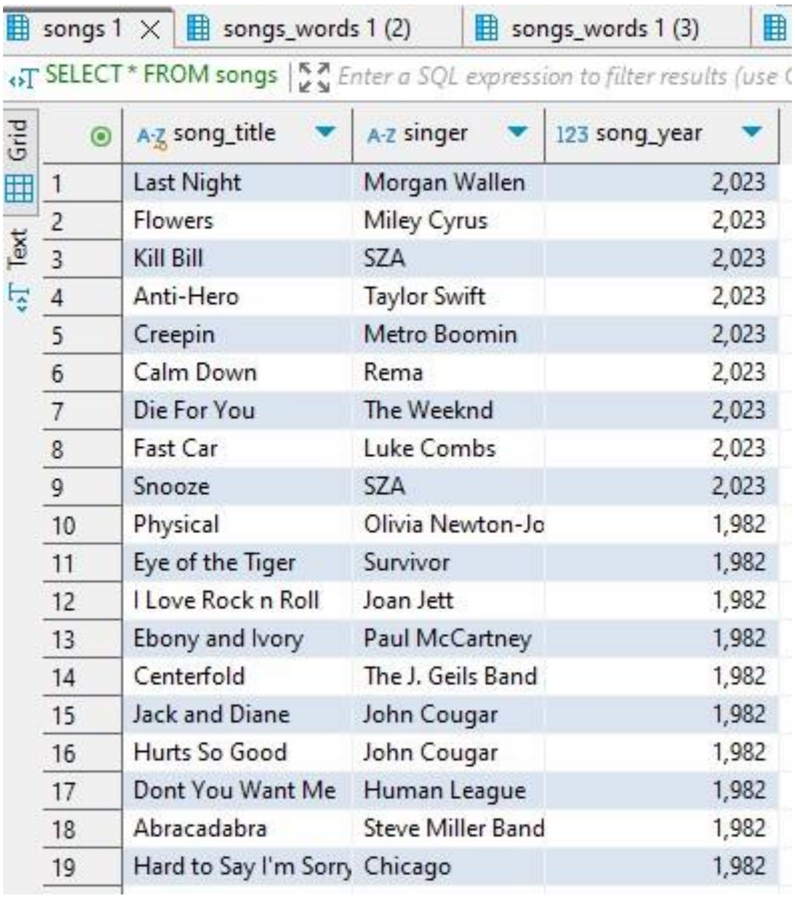
3. Find the top 5 most-used words for 1982:

```
SELECT word, SUM(word_count) AS total_count
FROM songs_words
```

```
JOIN songs ON songs_words.song_title = songs.song_title
WHERE song_year = 1982
GROUP BY word
ORDER BY total_count DESC
LIMIT 5;
```

TABLE STRUCTURES

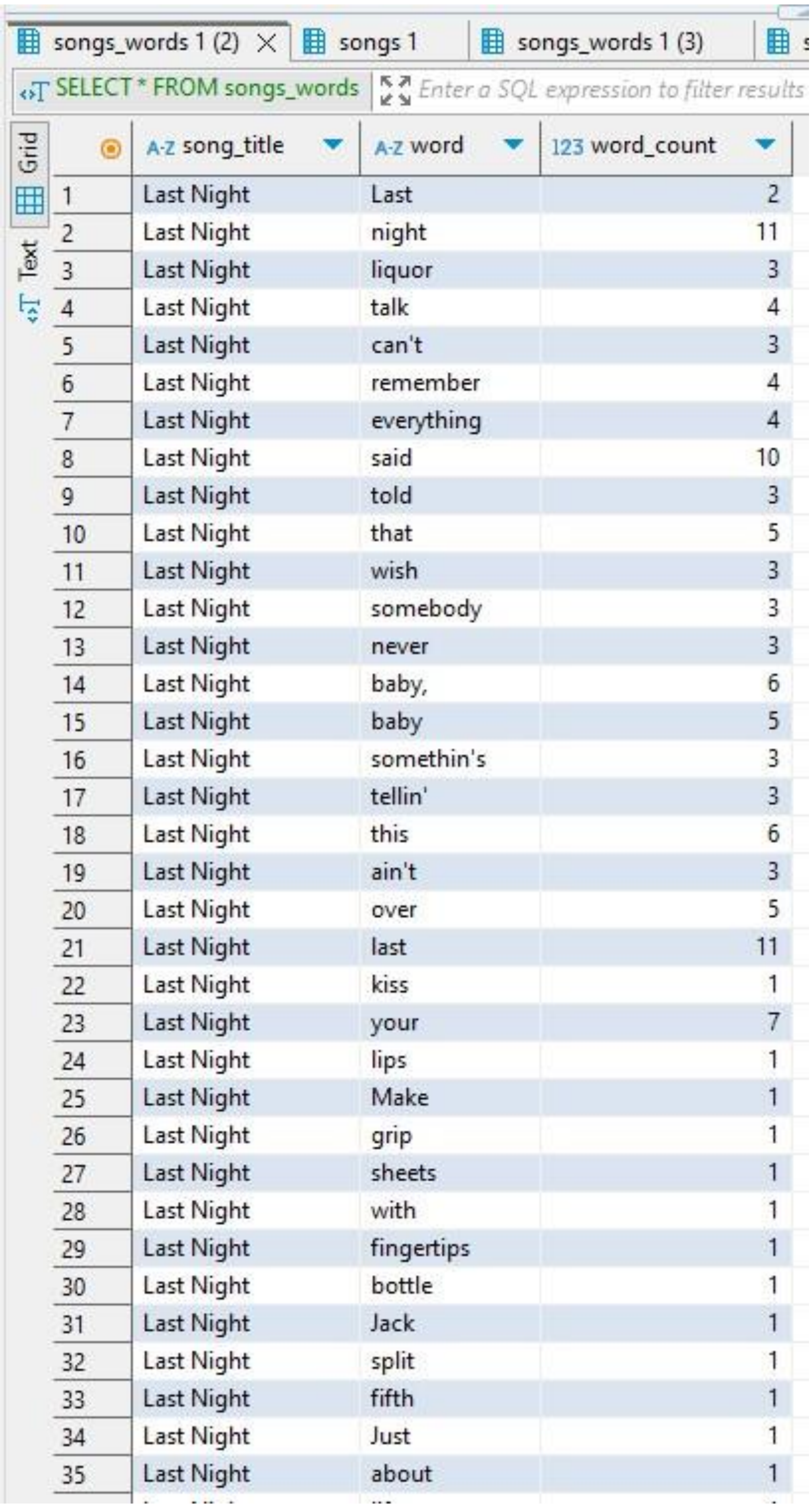
Songs Table:



```
SELECT * FROM songs
```

	song_title	singer	song_year
1	Last Night	Morgan Wallen	2,023
2	Flowers	Miley Cyrus	2,023
3	Kill Bill	SZA	2,023
4	Anti-Hero	Taylor Swift	2,023
5	Creepin	Metro Boomin	2,023
6	Calm Down	Rema	2,023
7	Die For You	The Weeknd	2,023
8	Fast Car	Luke Combs	2,023
9	Snooze	SZA	2,023
10	Physical	Olivia Newton-Jo	1,982
11	Eye of the Tiger	Survivor	1,982
12	I Love Rock n Roll	Joan Jett	1,982
13	Ebony and Ivory	Paul McCartney	1,982
14	Centerfold	The J. Geils Band	1,982
15	Jack and Diane	John Cougar	1,982
16	Hurts So Good	John Cougar	1,982
17	Dont You Want Me	Human League	1,982
18	Abracadabra	Steve Miller Band	1,982
19	Hard to Say I'm Sorry	Chicago	1,982

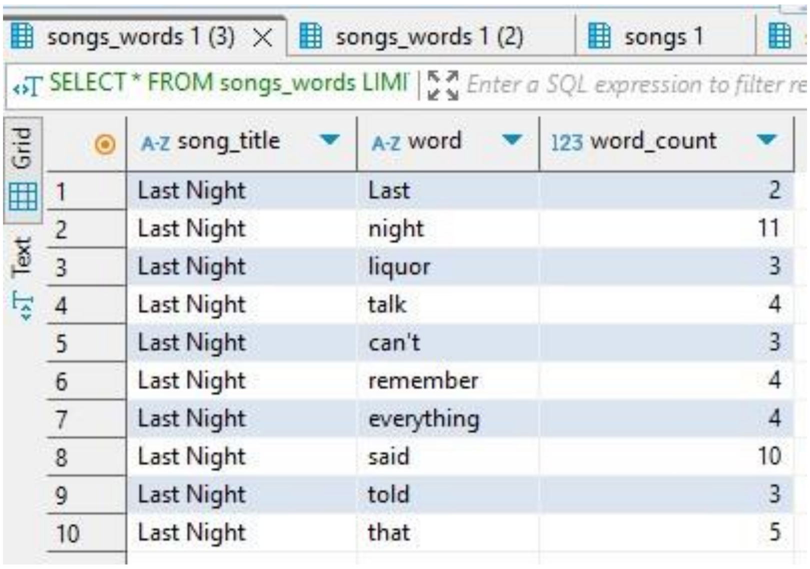
Songs_Words Table:



SELECT * FROM songs_words

	song_title	word	word_count
1	Last Night	Last	2
2	Last Night	night	11
3	Last Night	liquor	3
4	Last Night	talk	4
5	Last Night	can't	3
6	Last Night	remember	4
7	Last Night	everything	4
8	Last Night	said	10
9	Last Night	told	3
10	Last Night	that	5
11	Last Night	wish	3
12	Last Night	somebody	3
13	Last Night	never	3
14	Last Night	baby,	6
15	Last Night	baby	5
16	Last Night	somethin's	3
17	Last Night	tellin'	3
18	Last Night	this	6
19	Last Night	ain't	3
20	Last Night	over	5
21	Last Night	last	11
22	Last Night	kiss	1
23	Last Night	your	7
24	Last Night	lips	1
25	Last Night	Make	1
26	Last Night	grip	1
27	Last Night	sheets	1
28	Last Night	with	1
29	Last Night	fingertips	1
30	Last Night	bottle	1
31	Last Night	Jack	1
32	Last Night	split	1
33	Last Night	fifth	1
34	Last Night	Just	1
35	Last Night	about	1

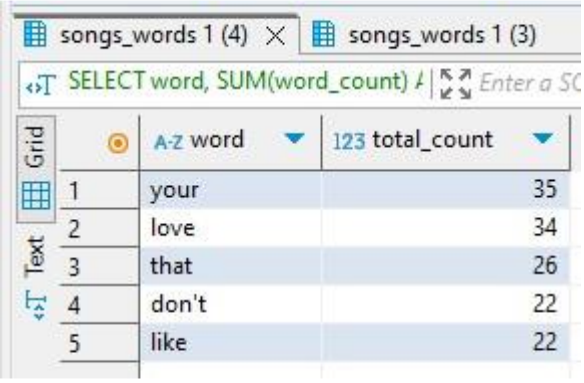
Songs_Words Table:



```
SELECT * FROM songs_words LIMI
```

	song_title	word	word_count
1	Last Night	Last	2
2	Last Night	night	11
3	Last Night	liquor	3
4	Last Night	talk	4
5	Last Night	can't	3
6	Last Night	remember	4
7	Last Night	everything	4
8	Last Night	said	10
9	Last Night	told	3
10	Last Night	that	5

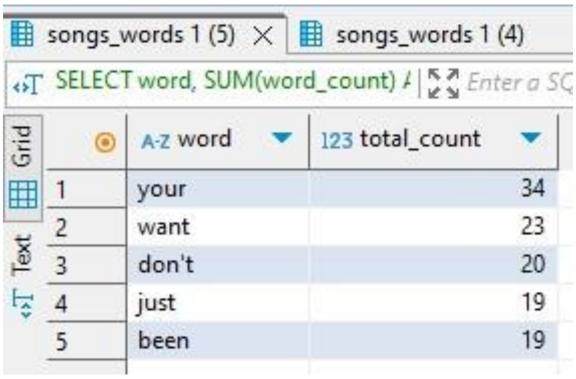
Songs_Words Table:



```
SELECT word, SUM(word_count) /
```

	word	total_count
1	your	35
2	love	34
3	that	26
4	don't	22
5	like	22

Songs_Words Table:



```
SELECT word, SUM(word_count) /
```

	word	total_count
1	your	34
2	want	23
3	don't	20
4	just	19
5	been	19

OBSERVATIONS

1. Results Overview:

Year	Top Words Identified	Themes Reflected
2023	*love, night, feel, heart, time*	Emotional, introspective themes
1982	*rock, heart, night, fight, fire*	Rebellious, energetic, passionate

2. Analysis:

2023 Songs - Emotional and Introspective Themes

- The frequent words such as "love," "night," and "feel" reflect a strong focus on personal emotions, relationships, and introspection.
- This mirrors the themes commonly found in today's pop, R&B, and alternative music genres, emphasizing personal struggles and emotional stories.

1982 Songs - Rebellious and Energetic Themes

- Words like "rock," "fight," and "fire" represent energy, defiance, and passion, typical of the 1980s rock and pop scene.
- The lyrics reflect societal attitudes of the time, focusing on personal freedom, resilience, and defiance against challenges.

3. Challenges:

1. **API Limitations:**
   - Some lyrics were unavailable due to API restrictions or missing data. For example, "Die For You" by The Weeknd had incomplete lyrics, requiring error-handling to avoid interruptions.
2. **Data Cleanup:**
   - Words shorter than 4 characters were removed to eliminate common stopwords and improve analysis accuracy.
3. **Data Conflicts:**
   - The SQL ON CONFLICT clause ensured no duplicate records were inserted, automatically updating word counts when needed.

CONCLUSION

This lab demonstrates how Python, PostgreSQL, and external APIs can work together to collect and analyse data. By processing lyrics into structured word frequency data, we gained insights into lyrical themes across decades. This approach has broad applications in fields like data analytics, natural language processing, and market research.