Project_Team04 Spotify Music Analyzing & Create a Playlist

組員 : 黃庭筠 112550105
謝欣陵 112550115
馬晨瑜 112550152
蕭宇岑 112550179
陳璽安 112550184

- Mainidea (5%)
  - The purpose of your application

    利用 Spotify API 獲得使用者喜好項目分類(語言、曲風、適合聽的時機、藝人、團體), 分析並且分類後撰寫插件, 將分類後的歌曲做成 filter 匯入 spotify 歌單。

- Data(15%)
  - Similar to what you did in the proposal, but this time we need a more complete version of description of your data
    - Tables, columns, data source constructed in your database, etc. your information about data should be as detail as possible

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_finalproject |
+------------------------+
| Album                  |
| AlbumStaging           |
| Singer                 |
| SingerStaging          |
| Song                   |
| SongStaging            |
| playlist               |
+------------------------+
7 rows in set (0.01 sec)
```

1. Album (data source: Album dataset)

```
mysql> DESCRIBE Album;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| album_name  | varchar(255) | NO   | PRI | NULL    |       |
| song_id     | varchar(50)  | NO   | PRI | NULL    |       |
| song_name   | varchar(255) | YES  |     | NULL    |       |
| singer_name | varchar(255) | NO   | PRI | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

2. Singer (data source: Spotify Artist Metadata Top 10k)

```
mysql> DESCRIBE Singer;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| singer_name   | varchar(255) | NO   | PRI | NULL    |       |
| singer_gender | varchar(10)  | YES  |     | NULL    |       |
| singer_age    | int          | YES  |     | NULL    |       |
| singer_type   | varchar(50)  | YES  |     | NULL    |       |
| singer_nation | varchar(50)  | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

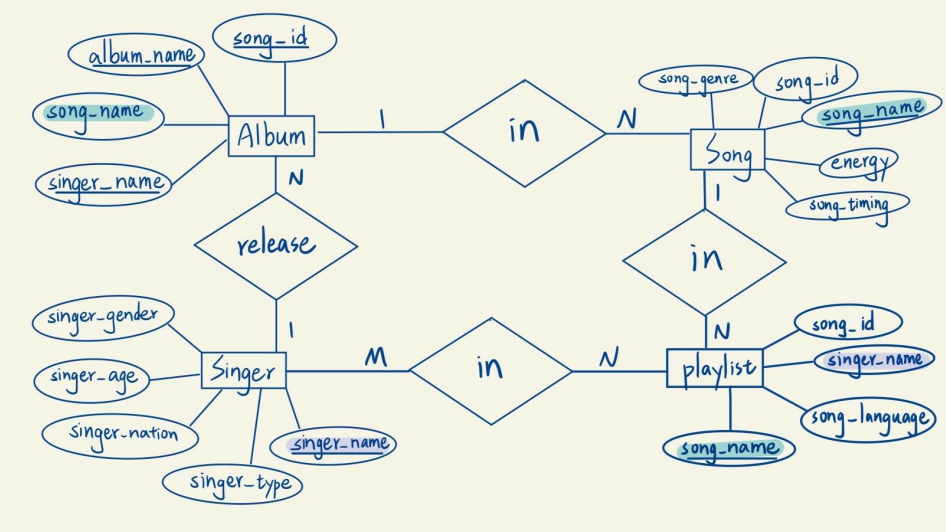
3. Song (data source: 🎹 Spotify Tracks Dataset)

```
mysql> DESCRIBE Song;
+-------------+--------------+------+-----+---------+-----------------+
| Field       | Type         | Null | Key | Default | Extra           |
+-------------+--------------+------+-----+---------+-----------------+
| song_id     | varchar(50)  | YES  |     | NULL    |                 |
| song_name   | varchar(200) | NO   | PRI | NULL    |                 |
| energy      | decimal(5,4) | YES  |     | NULL    |                 |
| song_timing | varchar(50)  | YES  |     | NULL    | STORED GENERATED |
| song_genre  | varchar(50)  | YES  |     | NULL    |                 |
+-------------+--------------+------+-----+---------+-----------------+
5 rows in set (0.01 sec)
```

4. playlist (data source: spotify API)

```
mysql> DESC playlist;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| song_name     | varchar(255) | NO   | PRI | NULL    |       |
| singer_name   | varchar(255) | NO   |     | NULL    |       |
| song_language | varchar(100) | NO   |     | NULL    |       |
| song_id       | varchar(255) | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

■ Draw an ER model of your schema



● Database (40%)

○ What database do you use (MySQL, SQLite, mongoDB, etc.)? MySQL

○ How do you maintain your database (update data, add new data etc.)

- 新增資料：使用SQL指令
  1. 暫存表 (Staging Table)：先將資料載入暫存表進行清理
  2. 目標表 (Target Table)：將清理後的資料插入正式的目標表
  3. 步驟：(這裡以Album 為例)
     (1) 建立資料表：使用 CREATE TABLE 指令建立正式表 (Album) 和暫存表 (AlbumStaging)

```sql
CREATE TABLE Album (
    album_name VARCHAR(255),
    song_id VARCHAR(50),
    song_name VARCHAR(255),
    singer_name VARCHAR(255),
    PRIMARY KEY(album_name, song_id, singer_name)
);

CREATE TABLE AlbumStaging (
    album_name VARCHAR(255),
    song_id VARCHAR(50),
    song_name VARCHAR(255),
    singer_name VARCHAR(255)
);
```

(2) 載入暫存表：使用 LOAD DATA INFILE 載入 CSV 檔中的資料

```
LOAD DATA INFILE '/var/lib/mysql-files/album.csv'
INTO TABLE AlbumStaging
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
```

(3) 格式化資料：透過 SET 指令轉換資料格式

```
(@id, @name, @dummy, @artists, @album, @dummy, @dummy, @dummy)
SET album_name = @album,
    song_id = @id,
    song_name = @name,
    singer_name = @artists;
```

(4) 插入正式表：將清理過的資料插入 Album，使用 INSERT IGNORE 確保避免重複插入

```
INSERT IGNORE INTO Album (album_name, song_id, song_name, singer_name)
SELECT album_name, song_id, song_name, singer_name
FROM AlbumStaging;
```

(5) 匯出資料：將正式表中的資料匯出為 CSV 檔

```
SELECT *
FROM Album
INTO OUTFILE '/var/lib/mysql-files/Album_output.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

○ Describe how you connect your database to your application as detail as possible

■ use text, graphs, tables, etc. make your explanation clear and complete

■ How does your back end process queries from applications? Do you do exception handling? What if someone does something unexpected?

○ Remember, this is the MOST IMPORTANT part of your project, please try to elaborate more about this part, describe all the details.

1. 使用Python 與 Flask 框架建構後端，並透過MySQL-Connector-python 連接資料庫
   - 配置參數

```
# Database Configuration
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'yourpassword',
    'database': 'finalproject'
}
```

   - 建立連線：使用 db_config 傳遞必要參數

```
# Database Connection
def get_db_connection():
    return mysql.connector.connect(**db_config)
```

2. 後端處理 query
   - 插入資料：
     - 使用 `reset_table` 函數重置資料表

- 例外處理：如果發生錯誤（例如檔案遺失或資料庫無法連線），會顯示對應的錯誤訊息

```
# Function to reset the table
def reset_table():
    try:
        connection = get_db_connection()
        cursor = connection.cursor()
        # Truncate the 'playlist' table to remove all rows and reset auto-increment
        cursor.execute("TRUNCATE TABLE playlist")
        connection.commit()
        print("Table 'playlist' has been reset.")
    except mysql.connector.Error as e:
        print(f"Database Error: {e}")
    finally:
        if 'connection' in locals() and connection.is_connected():
            connection.close()
```

- 查詢資料：
  - 使用`fetch_playlist_data`函數查詢資料並通過子查詢進行數據補充
  - 子查詢為 `playlist` 資料增加上下文資訊（例如歌手性別或歌曲類型）

```
# Fetch data from the 'playlist' table
def fetch_playlist_data():
    try:
        connection = get_db_connection()
        cursor = connection.cursor(dictionary=True)
        query='''...
        cursor.execute(query)
        return cursor.fetchall()
    except mysql.connector.Error as e:
        print(f"Database Error: {e}")
        return []
    finally:
        if 'connection' in locals() and connection.is_connected():
            connection.close()
```

- 例外處理：
  - 資料庫操作：連線失敗或 SQL 錯誤，錯誤時關閉連線

```
except mysql.connector.Error as e:
    print(f"Database Error: {e}")
finally:
    if 'connection' in locals() and connection.is_connected():
        connection.close()
```

  - API整合：

```
if 'error' in response_data:
    raise Exception(f"Spotify API Error: {response_data['error']}")
```

  - API 授權：

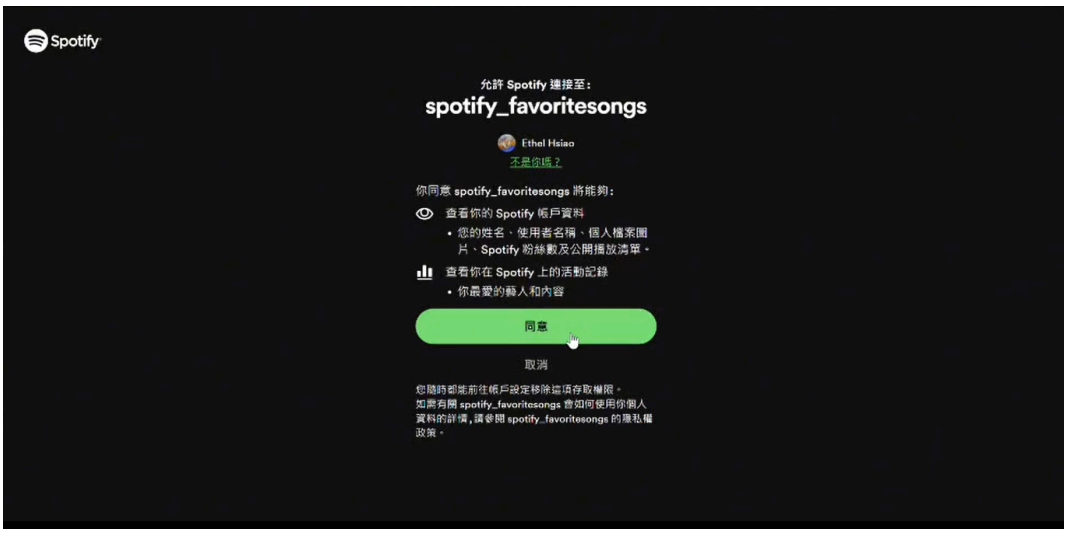
```
if 'access_token' not in response_data:
    raise Exception("No access token in response")
```

  - 使用者輸入：

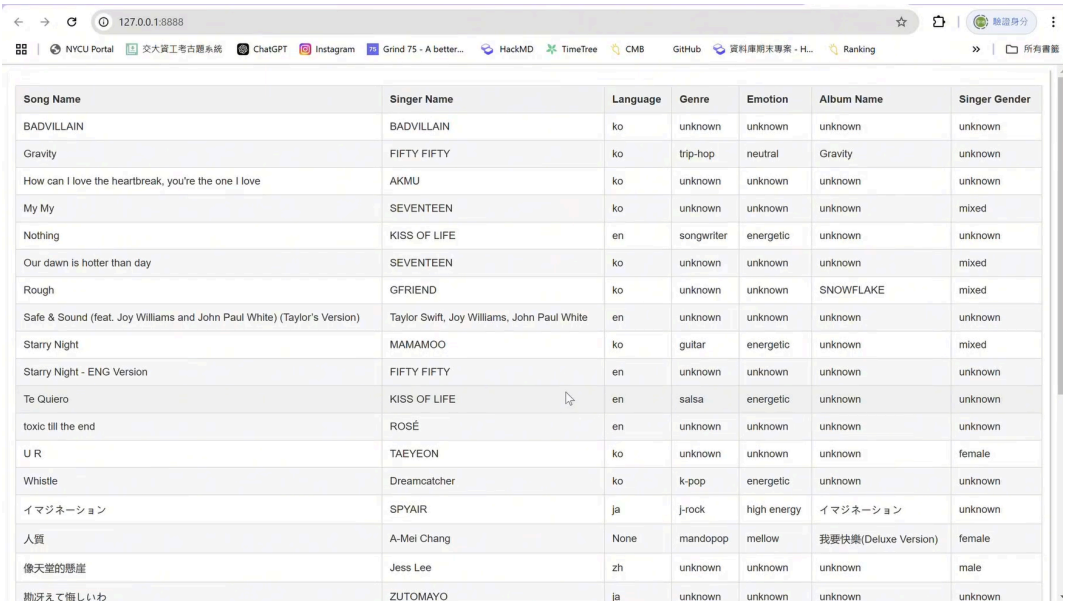
```
if not filtered_songs:
    return jsonify({'error': '沒有選擇任何歌曲'}), 400
```

- Application (30%)
  - Describe the interface of your application
    1. 登入

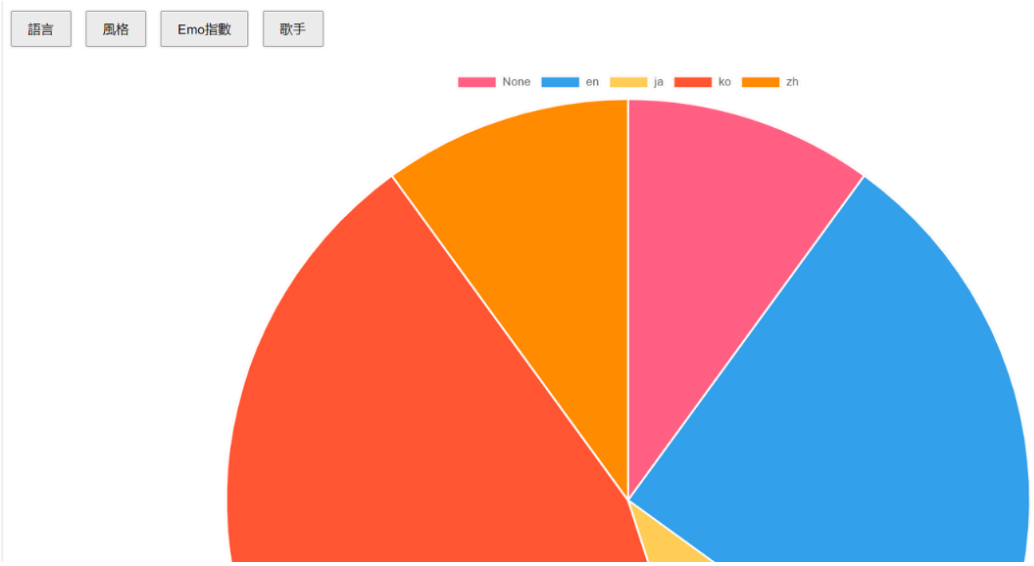
向Spotify API發送請求之後需要點擊授權便可以獲取使用者的toptracks(即近六個月最常聽的20首歌)，結合genius API判別歌曲的語言，生成playlist.csv(包括song_name、singer_name、song_language)



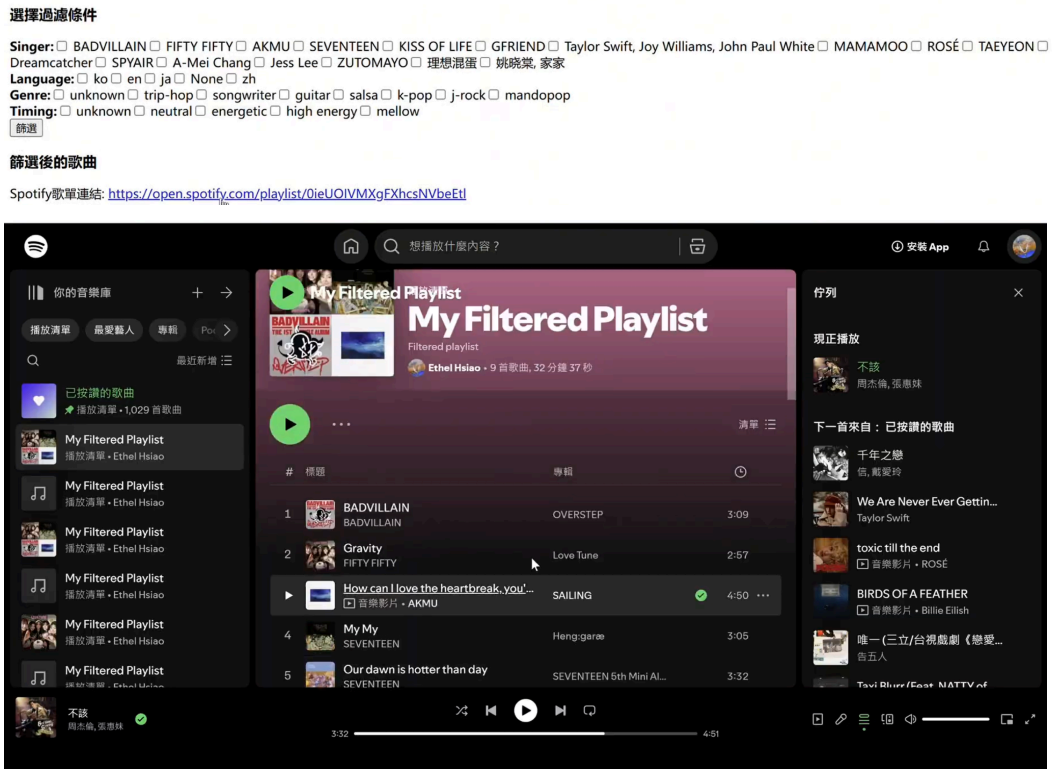
2. 分析
將playlist.csv的資料匯入Table playlist中，並且與結合已建立的Table Album、Song、Singer交叉搜尋，利用json將其資料連結至HTML表格內，可獲得自己top tracks的其他資訊(包含genre、emotion、album_name、singer_gender)，並且繪製圓餅圖分析。



Song Name	Singer Name	Language	Genre	Emotion	Album Name	Singer Gender
BADVILLAIN	BADVILLAIN	ko	unknown	unknown	unknown	unknown
Gravity	FIFTY FIFTY	ko	trip-hop	neutral	Gravity	unknown
How can I love the heartbreak, you're the one I love	AKMU	ko	unknown	unknown	unknown	unknown
My My	SEVENTEEN	ko	unknown	unknown	unknown	mixed
Nothing	KISS OF LIFE	en	songwriter	energetic	unknown	unknown
Our dawn is hotter than day	SEVENTEEN	ko	unknown	unknown	unknown	mixed
Rough	GFRIEND	ko	unknown	unknown	SNOWFLAKE	mixed
Safe & Sound (feat. Joy Williams and John Paul White) (Taylor's Version)	Taylor Swift, Joy Williams, John Paul White	en	unknown	unknown	unknown	unknown
Starry Night	MAMAMOO	ko	guitar	energetic	unknown	mixed
Starry Night - ENG Version	FIFTY FIFTY	en	unknown	unknown	unknown	unknown
Te Quiero	KISS OF LIFE	en	salsa	energetic	unknown	unknown
toxic till the end	ROSÉ	en	unknown	unknown	unknown	unknown
U R	TAEYEON	ko	unknown	unknown	unknown	female
Whistle	Dreamcatcher	ko	k-pop	energetic	unknown	unknown
イマジネーション	SPYAIR	ja	j-rock	high energy	イマジネーション	unknown
人質	A-Mei Chang	None	mandopop	mellow	我要快樂(Deluxe Version)	female
像天堂的懸崖	Jess Lee	zh	unknown	unknown	unknown	male
勘弁えて悔しいわ	ZUTOMAYO	ja	unknown	unknown	unknown	unknown



3. 創建歌單
   勾選自己所想要創建的歌單，按下篩選鍵發送請求給Spotify API，授權後即可取得建立歌單的連結，點擊後可查看在Spotify建立的歌單。



○ Describe the functions of your application (what it can do)

■ each of CRUD(create read update delete) function should be implemented at least one.

create：創建歌單

```
function createPlaylist() {
    const selectedSongs = [];
    document.querySelectorAll("#filteredSongs li").forEach(item => {
        selectedSongs.push(item.textContent);
    });

    localStorage.setItem("playlist", JSON.stringify(selectedSongs));
    window.location.href = "newplaylist.html";
}
```

read：檢視使用者近六個月最常聽的20首歌以及他們對應的singer_name、song_language、genre、timing、singer_gender

update：勾選不同類別的歌更新不同歌單

```
function filterSongs() {
    const selectedFilters = { language: [], singer: [], genre: [], timing: [] };
    document.querySelectorAll(".filter:checked").forEach(checkbox => {
        const type = checkbox.getAttribute("data-type");
        const value = checkbox.value;
        selectedFilters[type].push(value);
    });

    const rows = document.querySelectorAll("#songTable tbody tr");
    const filteredSongs = [];
    rows.forEach(row => {
        const cells = row.cells;
        const song = {
            song_name: cells[0].textContent.trim(),
            singer_name: cells[1].textContent.trim(),
            language: cells[2].textContent.trim(),
            genre: cells[3].textContent.trim(),
            timing: cells[4].textContent.trim()
        };

        const matchesFilter = (!selectedFilters.language.length || selectedFilters.language.includes(song.language)) &&
                              (!selectedFilters.singer.length || selectedFilters.singer.includes(song.singer_name)) &&
                              (!selectedFilters.genre.length || selectedFilters.genre.includes(song.genre)) &&
                              (!selectedFilters.timing.length || selectedFilters.timing.includes(song.timing));
        if (matchesFilter) filteredSongs.push(song.song_name);
    });
```

```
    const filteredSongsList = document.getElementById("filteredSongs");
    filteredSongsList.innerHTML = "";
    filteredSongs.forEach(song => {
        const li = document.createElement("li");
        li.textContent = song;
        filteredSongsList.appendChild(li);
    });

    const createPlaylistButton = document.getElementById("createPlaylistButton");
    createPlaylistButton.style.display = filteredSongs.length ? "block" : "none";
}
```

delete：在每一次新登入後，清除原先使用者的資料

```
# Function to reset the table
def reset_table():
    try:
        connection = get_db_connection()
        cursor = connection.cursor()
        # Truncate the 'playlist' table to remove all rows and reset auto-increment
        cursor.execute("TRUNCATE TABLE playlist")
        connection.commit()
        print("Table 'playlist' has been reset.")
    except mysql.connector.Error as e:
        print(f"Database Error: {e}")
    finally:
        if 'connection' in locals() and connection.is_connected():
            connection.close()
```

○ For each function of your application describe how you make it possible in detail

■ query design (including discussion on “why the query is designed like this”)

在最初設計查詢時，我們選擇使用 JOIN 作為連接資料表的方法，目的是要將 Playlist、Song 、Singer和Album 這四個資料表中的資訊做連結。然而在實作

過程中發現，由於並非所有在 Playlist 中的歌曲都會同時出現在 Song 、Singer 或Album中，使用一般的 JOIN 導致回傳的資料量大幅減少，無法完整呈現所有資訊。為了解決這個問題，我們原本打算改用 LEFT JOIN，這樣可以保留所有 Playlist 的資料，即使某些歌曲在其他Table中找不到對應的資訊。但是在實作時發現系統不支援 LEFT JOIN 函式，因此我們最終採取了另一個解決方案：修改查詢邏輯，當 Playlist 中的 singer_name 能在 Singer 資料表中找到對應時，就返回該歌手的性別資訊（Singer.singer_gender），如果找不到對應的資料，則返回 'Unknown' 作為預設值。這個方案既保留了所有播放清單的資料，也解決了系統限制的問題。(在Song、Album中則以song_name查詢)

```
query='''
    SELECT p.song_name, p.singer_name,
    CASE
        WHEN p.singer_name IN (SELECT singer_name FROM Singer) THEN
            (SELECT singer_gender FROM Singer WHERE Singer.singer_name = p.singer_name LI
        ELSE
            'unknown'
    END AS singer_gender,
    p.song_language,
    CASE
        WHEN p.song_name IN (SELECT song_name FROM Song) THEN
            (SELECT song_genre FROM Song WHERE Song.song_name = p.song_name LIMIT 1)
        ELSE
            'unknown'
    END AS song_genre,
    CASE
        WHEN p.song_name IN (SELECT song_name FROM Song) THEN
            (SELECT song_timing FROM Song WHERE Song.song_name = p.song_name LIMIT 1)
        ELSE
            'unknown'
    END AS song_timing,
    CASE
        WHEN p.song_name IN (SELECT song_name FROM Album) THEN
            (SELECT album_name FROM Album WHERE Album.song_name = p.song_name LIMIT 1)
        ELSE
            'unknown'
    END AS album_name
    FROM playlist p
'''
```

■ exception handling

若沒有連線到資料庫則返回”Database Error:”讓使用者知道資料庫未成功連線

```
except mysql.connector.Error as e:
    print(f"Database Error: {e}")
    return []# return empty list
```

- Others (10%)
  - The link to where your project code is stored (github/gitlab repository)
    https://github.com/tingyun1412/Database-Final-Project
  - The link to your video on YouTube
    https://youtu.be/Xta4iSIw7hg?si=5s8pI2CYruIfi_Oq
  - Draw the progress of your project
    - expected progress

time	progress

11/3~11/10	透過Spotify API獲得Spotify使用者的top-20-track
11/11~11/24	建立Database儲存獲得的Spotify歌單及其他資訊（Album, Song, Singer）
11/25~12/1	撰寫篩選歌單歌曲及建立Spotify歌單的插件
12/2~12/8	建立前端網頁

■ the actual progress

time	progress
11/3~11/10	後端 在Spotify Dashboard創建app, 利用Spotify access token得到使用者的top-20-tracks 歌詞語言偵測
11/11~11/17	後端 結合Spotfiy API和歌詞語言偵測，登入成功後後回傳Top-20-tracks的csv檔 Singer, Song, Album database建立 登入Spotfiy後拿到用於取得access token的驗證碼 前端 傳入csv檔可顯示使用者歌單（有歌名、歌手、語言欄位） 圓餅圖及點擊按鈕
11/18~11/24	後端 登入Spotify後可直接回傳使用者歌單的csv檔（處理完驗證碼及Access Token等銜接部分） 建立使用者歌單的table, 可將table送到前端顯示在畫面上 前端 可將傳入的使用者歌單照singer或language分類並繪製圓餅圖
12/25~12/30	後端 使用者登入Spotfiy獲取歌單後會直接傳進已建好的table，並將Table的資料傳入前端 用不同條件篩選歌曲 利用Spotfiy API搜尋使用者篩選歌單並建立Spotfiy歌單回傳歌單連結 兩介面連接（登入Spotfiy獲取歌單及篩選歌單介面） 前端 介面美化 不同篩選條件的圓餅圖 勾取篩選條件的checkbox 顯示歌單連結 利用Json連結前後端先在python檔案中選擇所需的資料並運用資料庫，再匯入HTML檔案智成圓餅圖以及進行其他分析。

○ What were the problems you met in the project, how did you solve them?
https://hackmd.io/@Huang05/rJuorEUeJg/edit
○ List the contribution of each team member in the project clearly

姓名	貢獻
黃庭筠	**Code** 建立Database、連結Database table 至 HTML **Presentation** 整體流程說明、interface.py、interface.html說明 **Report** Application
蕭宇岑	**Code** 取得top-20-tracks(favList_into_csv.py)、 判斷track語言(fuzzSearch.py)、 整合流程成start_process.py **Presentation** start_process.py , favList_into_csv.py的function 說明 **Report**
陳璽安	**Code** 使用者登入後獲取授權碼、取得使用者top-20-track的csv檔、利用spotify API搜尋使用者篩選歌單、創建新歌單並顯示連結、連接取得歌單與創建歌單兩介面 **Presentation** interface.py, interface.html內程式碼詳細說明 **Report** progress of project
馬晨瑜	**Code** 最初的interface.html interface___.html (本來就是備案)、newplaylist.html (最後沒用到) **Presentation** Main idea (目的、動機、未來可能發展方向) **Report** Main idea、data、database
謝欣陵	**Code** 建立Database **Presentation** Database、Dataset和ER model的詳細說明 **Report**